OSCAR: THE OCCAM'S RAZOR FOR EXTREME KV CACHE QUANTIZATION IN LLMS AND BEYOND

Anonymous authors
Paper under double-blind review

ABSTRACT

The rapid advancement toward long-context reasoning and multi-modal intelligence has made the memory footprint of the Key-Value (KV) cache a dominant memory bottleneck for efficient deployment. Extreme low-bit quantization has emerged as a fundamental imperative to reclaim memory efficiency and sustain high-throughput inference. While the established per-channel quantization effectively accommodates intrinsic channel-wise outliers in Key tensors, its efficacy diminishes under extreme compression. In this work, we revisit the inherent limitations of the per-channel quantization paradigm from both empirical and theoretical perspectives. Our analysis identifies *Token Norm Imbalance (TNI)* as the primary bottleneck to quantization fidelity. We demonstrate that TNI systematically amplifies errors when shared quantization parameters are required to span token groups exhibiting substantial norm disparities. Instead of relying on intricate quantization pipelines (e.g., TurboQuant), we propose *OScaR (**O**mni-**Sca**led Canalized **R**otation)*, an accurate and lightweight KV cache compression framework for X-LLMs (i.e., text-only, multi-modal, and omni-modal LLMs). Advancing the per-channel paradigm, OScaR employs *Canalized Rotation* followed by *Omni-Token Scaling* to mitigate TNI-induced sequence-dimensional variance both effectively and efficiently, further supported by our optimized system design and CUDA kernels. Extensive evaluations across X-LLMs show that OScaR consistently outperforms existing methods and achieves near-lossless performance under INT2 quantization, establishing it as a robust, low-complexity, and universal framework that defines a new Pareto front. Compared with the `BF16 FlashDecoding-v2` baseline, our OScaR implementation achieves a notable up to 3.0× speedup in decoding, reduces memory footprint by 5.3×, and increases throughput by 4.1×. Code for result reproduction is provided in the anonymous supplementary materials.

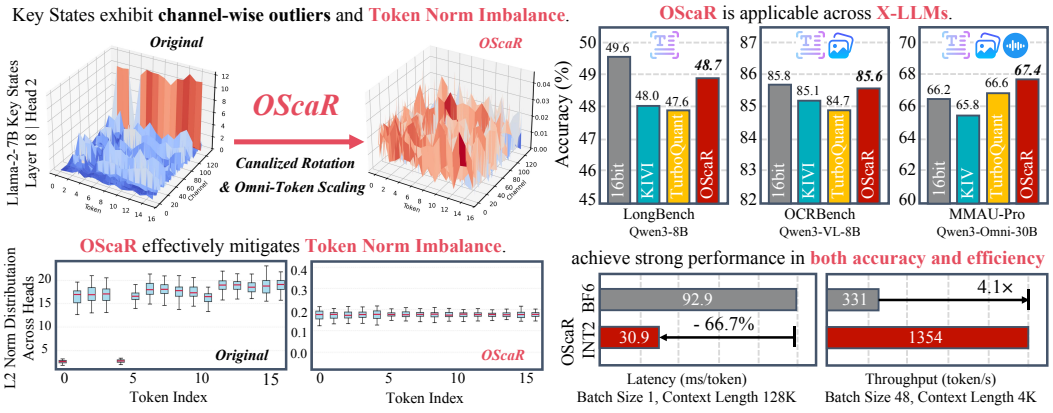


Figure 1: Conceptual overview of this paper. We revisit the per-channel key quantization paradigm and identify its inherent limitation, termed *token norm imbalance (TNI)*. We then propose OScaR, a streamlined framework that applies *Canalized Rotation* followed by *Omni-Token Scaling* to effectively mitigate TNI. Extensive evaluations across X-LLMs demonstrate that OScaR establishes a superior accuracy-efficiency Pareto front.

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

1 INTRODUCTION

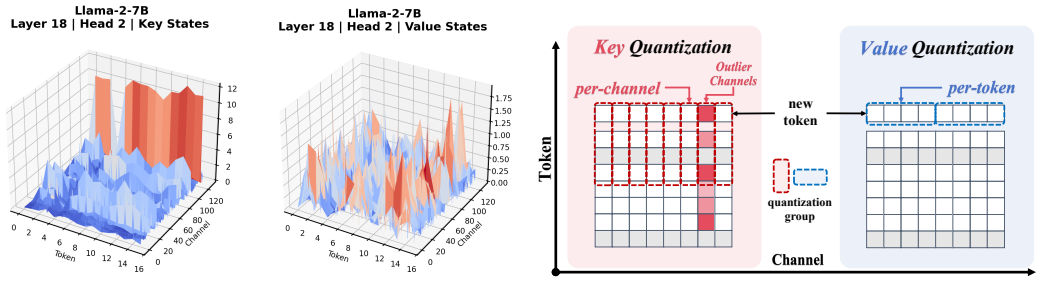
Recent advancements in large language models (LLMs) and their multi-modal counterparts have demonstrated remarkable capabilities in complex reasoning and multi-modal perception (Meituan LongCat Team et al., 2025a;c; 2026a; 2025d;b), establishing a new foundation for artificial intelligence. To further unlock these emergent abilities, the research frontier is increasingly prioritizing long-context processing, streaming tasks, and long-range audio-video multi-modal understanding (Wang et al., 2026; Meituan LongCat Team et al., 2025b;e; 2026b). However, these trends necessitate handling massive context sequences, causing the memory footprint of the Key-Value (KV) cache to grow linearly and dominate total memory consumption (Li et al., 2025a; Liu et al., 2025). In memory-bound inference scenarios, the KV cache rapidly exhausts the High Bandwidth Memory (HBM) capacity of modern accelerators, severely restricting batch sizes and hindering efficient large-scale deployment (Liu et al., 2024d; Hooper et al., 2024; Ge et al., 2023; Liu et al., 2024a). Consequently, reclaiming memory efficiency while sustaining high-throughput inference has become a fundamental imperative for next-generation LLMs (Kimi Team et al., 2025; Cao et al., 2026; Meituan LongCat Team et al., 2026b; 2025e).

To address these constraints, KV cache compression has matured into a significant research frontier, with methodologies such as quantization, pruning, and low-rank decomposition being extensively explored (Liu et al., 2024d; Hooper et al., 2024; Liu et al., 2024a; Ge et al., 2023; Wan et al., 2024; Cai et al., 2024). By mapping high-precision tensors to reduced bit-widths, quantization reduces memory overhead without compromising the structural integrity of the KV cache (Li et al., 2025a; Liu et al., 2024d; Hooper et al., 2024). Within the landscape of KV cache quantization, Key quantization has emerged as a focal point, posing more substantial challenges than Value quantization due to salient channel-wise outliers (Liu et al., 2024d; Hooper et al., 2024; Su et al., 2025c; Jin et al., 2025). Specifically, a sparse subset of channels within Key tensors often exhibits disproportionately large magnitudes. To mitigate this, per-channel Key quantization, which leverages intrinsic distributional characteristics, has proven to be a promising approach (Liu et al., 2024d; Hooper et al., 2024; Su et al., 2025a; Tao et al., 2025; Su et al., 2026b; Zandieh et al., 2025b) (see also Appendix B).

Although the per-channel quantization paradigm has achieved notable success, its effectiveness progressively diminishes under extreme compression (Liu et al., 2024d; Duanmu et al., 2024; Su et al., 2025c; Zandieh et al., 2025a). In this study, we revisit the inherent limitations of per-channel quantization. Through a meticulous token-wise norm distribution analysis of KV caches across multiple text-only and multi-modal LLMs, we identify a pervasive structural property, which we term *Token Norm Imbalance (TNI)*. Intuitively, TNI undermines per-channel quantization because shared quantization parameters must accommodate token groups with highly divergent norms (Nagel et al., 2021). Our empirical validation confirms that TNI systematically amplifies quantization error. Going beyond empirical exploration, our theoretical analysis further corroborates TNI-induced error amplification within per-channel quantization, revealing TNI as a fundamental vulnerability of the per-channel paradigm.

Existing KV cache quantization methods often lean heavily on auxiliary mechanisms to suppress quantization errors (Zandieh et al., 2025a; Pope, 2026; Zandieh et al., 2025b; Han et al., 2025). These intricate pipelines, coupled with unavoidable on-the-fly quantization, introduce substantial computational overhead and extra parameters, undermining practical viability. **Guided by the principle of Occam's Razor, we advocate for elegance and simplicity over intricate, heavy-weight quantization pipelines.** To this end, we introduce *OScaR (**O**mni-**Sca**led Canalized **R**otation)*, an accurate and lightweight KV cache quantization framework designed for X-LLMs (i.e., text-only, multi-modal, and omni-modal LLMs). As discussed in Section 3.2, building upon the established per-channel paradigm, OScaR first applies the Hadamard transform to prevent *Scaling-Induced Outlier Artifacts* from biasing the subsequent token scaling process (*Canalized Rotation*). Subsequently, *Omni-Token Scaling* performs omnidirectional sequence-level normalization to effectively mitigate the impact of diverse TNI patterns. The resulting pipeline remains training-free and highly streamlined, with both components being mutually essential.

Our empirical evaluations, along with theoretical complexity analyses across a diverse set of representative methods, demonstrate that OScaR's methodology is both robust and computationally efficient. Moreover, OScaR is built upon our carefully optimized system design and CUDA kernels, ensuring hardware efficiency and immediate deployability. Figure 1 provides a comprehensive overview of our paper. **The main contributions of our work are summarized as follows:**



(a) Key distribution. (b) Value distribution. (c) Schematic illustration of KIVI.

Figure 2: Key and Value magnitude patterns and the KIVI quantization scheme (Liu et al., 2024d). Key states exhibit significant channel-wise outliers, necessitating per-channel quantization. In contrast, Value states have a relatively uniform magnitude distribution and are quantized per-token.

- **Unveiling TNI as the Structural Bottleneck of Per-Channel Quantization:** We identify Token Norm Imbalance (TNI) as the fundamental bottleneck limiting per-channel quantization in X-LLMs, supported by empirical evaluations and theoretical analysis.
- **Streamlined OScaR Framework:** Guided by the principle of Occam's Razor, we introduce OScaR, an accurate and lightweight KV cache quantization framework for X-LLMs. It first applies *Canalized Rotation* to prevent *Scaling-Induced Outlier Artifacts*, followed by *Omni-Token Scaling* to safely mitigate the impact of TNI.
- **Redefining the Pareto Front:** Extensive evaluations across X-LLMs demonstrate that OScaR outperforms existing methods while achieving near-lossless performance under INT2 quantization. By preserving high quantization fidelity and maintaining low overall complexity, OScaR establishes an advantageous accuracy-efficiency Pareto front.
- **Optimized CUDA Implementations and Efficiency Gains:** We provide a carefully optimized system design and dedicated CUDA kernels that translate theoretical insights into tangible performance improvements. Compared with the `BF16 FlashDecoding-v2` baseline, our implementation achieves up to a 3.0× decoding speedup, reduces memory footprint by 5.3×, and increases inference throughput by 4.1×.

2 Preliminaries

2.1 KV Caching in Autoregressive Inference

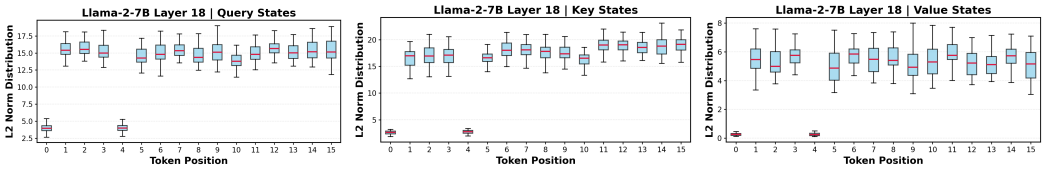
LLMs predominantly employ a Transformer decoder-only architecture, where KV caching eliminates redundant computations during autoregressive decoding (Vaswani et al., 2017; Liu et al., 2025; Li et al., 2025a). In multi-modal configurations, the LLM backbone integrates heterogeneous tokens from modality-specific encoders, projecting them into a shared latent space (Meituan LongCat Team et al., 2025b;e; Liu et al., 2023; 2024b). During the *prefill stage*, textual tokens $\mathbf{X}_T$, visual features $\mathbf{X}_V$, and audio embeddings $\mathbf{X}_A$ are concatenated along the sequence dimension to form the prompt sequence $\mathbf{X}_{\text{prompt}} = [\mathbf{X}_T; \mathbf{X}_V; \mathbf{X}_A] \in \mathbb{R}^{S\times D}$, where S is the total sequence length and D the hidden dimension. For each Transformer layer $l \in \{1, \dots, \mathcal{L}\}$, the hidden state $\mathbf{H}^{(l-1)}$ is linearly projected to obtain the Key and Value states forming the initial KV cache:

$$K^{(l)} = \mathbf{H}^{(l-1)} W_K^{(l)}, \quad V^{(l)} = \mathbf{H}^{(l-1)} W_V^{(l)}, \tag{1}$$

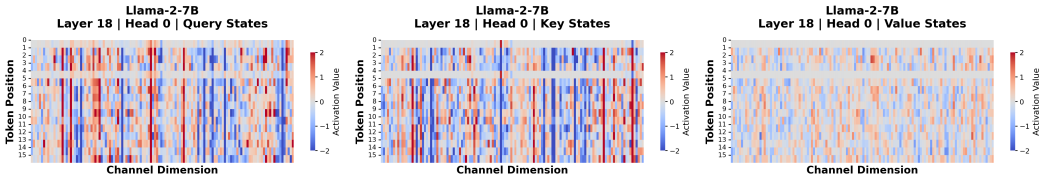
where $\mathbf{H}^{(0)} = \mathbf{X}_{\text{prompt}}$, and $W_K^{(l)}, W_V^{(l)} \in \mathbb{R}^{D\times D_{\text{kv}}}$ are Key/Value projection weights, with $D_{\text{kv}} = H_{\text{kv}} d_h$ for H_{kv} KV heads of dimension d_h. During the *decoding stage*, for each layer $l \in \{1, \dots, \mathcal{L}\}$ and step t, the input $\mathbf{h}_t^{(l-1)} \in \mathbb{R}^{1\times D}$ is projected to produce the Query, Key, and Value vectors:

$$\mathbf{q}_t^{(l)} = \mathbf{h}_t^{(l-1)} W_Q^{(l)}, \quad \mathbf{k}_t^{(l)} = \mathbf{h}_t^{(l-1)} W_K^{(l)}, \quad \mathbf{v}_t^{(l)} = \mathbf{h}_t^{(l-1)} W_V^{(l)}. \tag{2}$$

The KV cache for each layer is updated by concatenating the new vectors: $K^{(l)} \leftarrow [K^{(l)}; \mathbf{k}_t^{(l)}]$, $V^{(l)} \leftarrow [V^{(l)}; \mathbf{v}_t^{(l)}]$. The KV cache memory footprint grows linearly with sequence length, creating a memory-bound bottleneck that motivates compression.



(a) Query L2 norm distribution (b) Key L2 norm distribution (c) Value L2 norm distribution



(d) Query heatmap (e) Key heatmap (f) Value heatmap

Figure 3: L2 norm distributions (top row) and heatmaps (bottom row) of Query, Key, and Value states. Each attention state contains a sparse yet consistent subset of tokens with exceptionally low norms.

2.2 Block-Wise Per-Channel Quantization

Key states exhibit significant channel-wise outliers, while Value states have a relatively uniform magnitude distribution, as shown in Figure 2. Exploiting these distinct numerical distributions, a range of approaches adopt a hybrid quantization scheme that applies per-channel quantization to Keys while preserving per-token granularity for Values (Liu et al., 2024d; Su et al., 2025a; 2026b; Hooper et al., 2024). To integrate per-channel quantization into token-wise LLM decoding, the pioneering KIVI framework introduces a block-wise per-channel quantization strategy for the Key cache (Liu et al., 2024d). Specifically, given a Key cache $\mathbf{K} \in \mathbb{R}^{S\times d}$, where S denotes the sequence length and d the head dimension, each channel is partitioned into consecutive blocks of size G for quantization. For the j-th channel within block g, the quantization step size $\Delta_{j,g}$ and zero-point $z_{j,g}$ are computed as:

$$\Delta_{j,g} = \frac{\max_{i\in g} K_{i,j} - \min_{i\in g} K_{i,j}}{2^b - 1}, \qquad z_{j,g} = \left\lfloor -\frac{\min_{i\in g} K_{i,j}}{\Delta_{j,g}} \right\rceil. \tag{3}$$

Each element $K_{i,j}$ is then quantized and reconstructed as:

$$Q(K_{i,j}) = \text{clamp}\left(\left\lfloor \frac{K_{i,j}}{\Delta_{j,g}} \right\rceil + z_{j,g},\, 0,\, 2^b - 1\right), \qquad \hat{K}_{i,j} = \Delta_{j,g} \cdot \big(Q(K_{i,j}) - z_{j,g}\big). \tag{4}$$

Importantly, a high-precision residual window mechanism is required to support continuous per-channel quantization during autoregressive generation: newly generated tokens are appended to this buffer, maintained in full precision, and block-wise quantized only once the buffer accumulates the predefined residual number R. Background on low-bit quantization is provided in Appendix D.

3 Methodology

This section is organized into three parts. Section 3.1 revisits the inherent limitations of per-channel quantization and establishes *Token Norm Imbalance* as the fundamental bottleneck. Section 3.2 introduces OScaR and its algorithmic design, which comprises *Canalized Rotation* followed by *Omni-Token Scaling*. Section 3.3 presents our efficient system design and CUDA implementations.

3.1 Revisiting Per-Channel Key Quantization

While per-channel quantization mitigates channel-wise outliers, it inherently assumes that tokens within a given channel share similar magnitudes. When the within-channel distribution becomes skewed or contains even a few divergent tokens, the shared quantization parameters for that block are severely compromised, causing substantial fidelity degradation (Nagel et al., 2021). In this subsection, we systematically examine this assumption through (i) empirical observations, (ii) theoretical derivations, and (iii) quantitative error analysis.

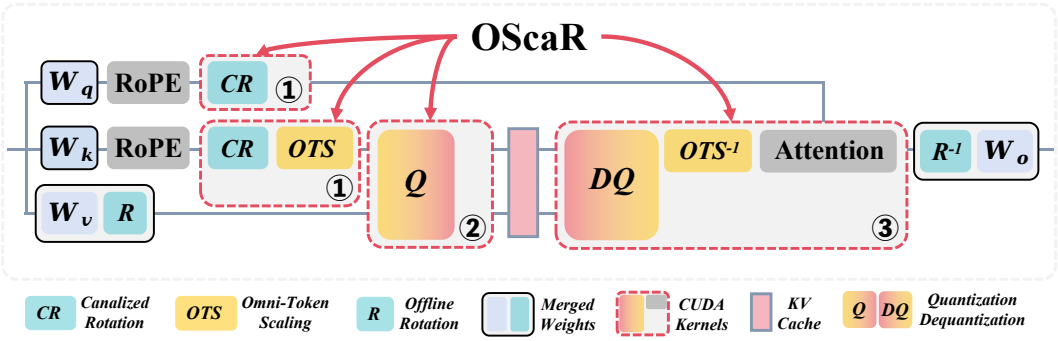


Figure 4: Conceptual overview of OScaR. The detailed algorithm is presented in Algorithm 1.

Empirical Observations. Our analysis is conducted across multiple mainstream open-source LLMs and multi-modal LLMs with fixed inputs (e.g., prompts, images). Systematic token-wise norm distribution profiling of KV caches consistently reveals substantial inter-token norm disparity, which we term *Token Norm Imbalance (TNI)*. Specifically, our experimental procedure is as follows. For each token position t in a transformer layer, we compute its ℓ_2 norm across all attention heads for the Query, Key, and Value states. These head-wise norms are aggregated into the set

$$\mathcal{N}_t^{(M)} = \left\{ \|\mathbf{t}_{t,h}^{(M)}\|_2 \;\middle|\; h = 1, \ldots, H \right\}, \quad \|\mathbf{t}_{t,h}^{(M)}\|_2 = \sqrt{\sum_{j=1}^{d_h} \left(s_{t,h,j}^{(M)}\right)^2}, \tag{5}$$

where d_h is the head dimension and $s_{t,h,j}^{(M)}$ denotes the j-th component of the token vector in head h for state $M \in \{\text{Query}, \text{Key}, \text{Value}\}$. The set $\mathcal{N}_t^{(M)}$ captures token variation across attention heads and serves as the basis for boxplot visualizations, where each token is represented by a single box illustrating the distribution of its head-wise norms.

Visualizations based on Llama-2-7B are shown in Figure 3. Additional results for text-only LLMs (Llama-3.1-8B, Qwen-3-8B) and the prompt used are provided in Appendix E. These results reveal significant outlier tokens as a manifestation of TNI. Specifically, each attention state contains a sparse yet consistent subset of tokens with exceptionally low norms. Their presence expands the quantization dynamic range for the corresponding block, representing the weakest link in the per-channel paradigm. Moreover, these low-norm outlier tokens consistently appear across different attention states and correspond directly to Attention Sink tokens (Su et al., 2026c; Xiao et al., 2023b), aligning with prior findings (Su & Yuan, 2025). Appendix F discusses Attention Sink tokens as low-norm outliers.

Beyond text-only LLMs, extensive TNI observations also hold in multi-modal LLMs. In such settings, TNI manifests not only as attention-sink-related outlier tokens but also through several distinct patterns: (i) broader token norm variation relative to text-only LLMs (Figure 19); (ii) inter-modality norm disparities, wherein norms remain smooth within each modality yet diverge substantially across modalities (Figure 20); and (iii) exceptionally large-norm outlier tokens, which contrast with the low-norm Attention Sink (Figure 21). Appendix G provides representative visualizations.

Theoretical Derivations. Building on the empirical observations of TNI across X-LLMs, we provide theoretical derivations of TNI-induced errors in per-channel quantization. Detailed derivations are presented in Appendix H, with an attention-logit analysis in Appendix H.1. As shown in Equation 11, the reconstruction error of a per-channel quantization block is fundamentally governed by the range of token norms within the block. Thus, TNI systematically amplifies quantization errors, revealing TNI as a fundamental vulnerability of the per-channel paradigm.

Quantitative Error Analysis. We conduct an empirical quantization error analysis under extreme KV cache compression to comprehensively quantify the impact of TNI. As shown in Table 2, TNI significantly affects per-channel Key quantization. For per-token Value quantization, although TNI persists, per-token quantization confines norm variations to individual tokens and avoids cross-token interference. Consequently, the error amplification caused by TNI in per-channel schemes does not manifest under per-token quantization. These analysis results validate our assumption and theoretical derivations. Further details and real-tensor calibration appear in Appendices I and I.1.

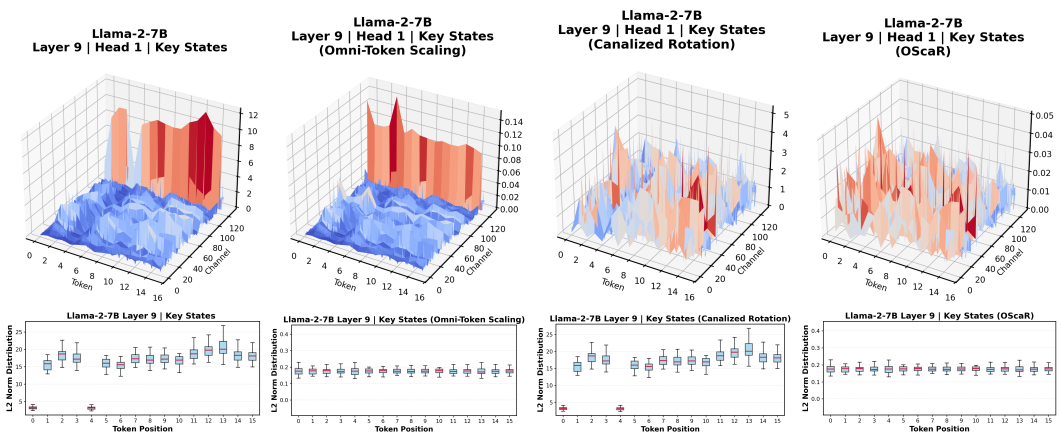


Figure 5: Key magnitude (top row) and L2 norm distribution (bottom row) across different processing stages: Original, after *Omni-Token Scaling* alone, after *Canalized Rotation* alone, and the full OScaR. Direct scaling balances token norms but introduces the *Scaling-Induced Outlier Artifact*. *Canalized Rotation* alone fails to balance token norms. Only the complete OScaR successfully addresses TNI without incurring the artifact. Additional visualizations are provided in Appendix K.

3.2 The OScaR Framework: Omni-Scaled Canalized Rotation

In this section, we introduce *OScaR (**O**mni-**Sca**led Canalized **R**otation)*, an accurate and lightweight KV cache compression framework for X-LLMs (i.e., text-only, multi-modal, and omni-modal LLMs). We focus on the algorithmic design herein, while the optimized system design and CUDA kernels are presented in the next subsection. An overview of the OScaR pipeline is provided in Figure 4, and the detailed algorithm is given in Algorithm 1.

Advancing the per-channel paradigm, OScaR introduces two key innovations that together mitigate TNI-induced sequence-dimensional variance in a fully training-free manner:

- **Canalized Rotation:** Direct token-wise scaling, though conceptually straightforward, suffers from the *Scaling-Induced Outlier Artifact* in practice. Applying Canalized Rotation prior to scaling suppresses outlier channels that would otherwise dominate token norms, thereby preventing this artifact from biasing subsequent *Omni-Token Scaling*.
- **Omni-Token Scaling:** Addresses TNI through omni-directional sequence-level normalization. Following *Canalized Rotation*, it safely applies token-wise scaling to balance token norms across the sequence dimension, thereby resolving the impact of diverse TNI patterns.

The effectiveness of these two components is demonstrated in Figure 5. Guided by Occam's Razor, OScaR avoids complex auxiliary pipelines and instead relies on the two mutually essential components described above to effectively and efficiently mitigate TNI. Below, we detail the design rationales and specific methodologies of the OScaR framework.

On the Failure of Direct Token-Wise Scaling A straightforward strategy to mitigate TNI is to apply token-wise scaling directly. However, although it balances token norms, empirical evidence shows that this approach rarely improves quantized models and, in many cases, even leads to degradation. Our analysis attributes this failure to what we term *Scaling-Induced Outlier Artifact*. Intuitively, consider normal tokens dominated by outlier channels and low-norm outlier tokens with relatively uniformly small entries. When scaled to the same norm, the low-norm tokens are uniformly amplified and become artificial outliers in channels where normal tokens have minimal magnitudes, expanding the per-channel quantization range and degrading precision. This artifact undermines per-channel quantization and cannot be resolved by merely adjusting the scaling target. Therefore, direct token-wise scaling alone is insufficient for handling TNI. A detailed analysis of *Scaling-Induced Outlier Artifact* is provided in Appendix J.

Canalized Rotation Followed by Omni-Token Scaling To resolve the *Scaling-Induced Outlier Artifact*, OScaR introduces a two-step procedure. First, *Canalized Rotation* applies a Hadamard transform to redistribute the energy of outlier channels across all dimensions. Second, *Omni-Token Scaling* computes the ℓ_2 norm of each token across all modalities and applies token-wise scaling to unify these norms. Because *Canalized Rotation* has already smoothed the per-channel distribution, the scaling step can safely balance token norms without introducing artificial outliers. As shown in Figure 5, rotation alone is insufficient for handling TNI. Only when combined with scaling can the framework effectively address TNI while avoiding the *Scaling-Induced Outlier Artifact*.

Occam's Razor for Extreme KV Cache Quantization Existing KV cache quantization methods typically rely on online quantization operations augmented with auxiliary mechanisms to mitigate errors (Zandieh et al., 2025a;b; Pope, 2026; Ji, 2026; Han et al., 2025) (see Appendix J.1 for a comparison). These complex pipelines incur substantial computational overhead and additional parameters, limiting both practicality and efficiency. A viable solution to TNI must therefore be concise and essential, ensuring high efficiency in real-world deployments. **Guided by the principle of Occam's Razor, we advocate simplicity and elegance over intricate, heavyweight quantization pipelines.** Theoretical complexity analysis in Appendix L demonstrates that OScaR is a highly lightweight approach. Combined with comprehensive benchmarks and efficiency evaluations in Section 4, OScaR achieves a clearly advantageous position on the accuracy-efficiency Pareto front.

3.3 Efficient System Design and CUDA Implementations of OScaR

OScaR Pipeline Overview As illustrated in Figure 4 and detailed in Algorithm 1, OScaR proceeds as follows. (i) For the Query, an online Fast Hadamard Transform (FHT) is applied to implicitly cancel the FHT applied to the Key during attention computation. (ii) The Key undergoes an online FHT followed by token-wise norm scaling, after which per-channel quantization is applied. During dequantization, inverse token-wise scaling restores the original token norms. (iii) For the Value, an offline Hadamard transform is applied to both the Value and the attention output weight matrices prior to inference. Per-token quantization is then applied during inference. This offline transform improves the fidelity of Value quantization without introducing additional runtime overhead (Ashkboos et al., 2024; Su et al., 2025c).

System Design and CUDA Kernels OScaR is implemented using three CUDA kernels, building upon `HadaCore` and `BitDecoding` (Agarwal et al., 2024; Du et al., 2025) with carefully engineered adaptations for high-performance execution on GPU Tensor Cores: (i) *Online FHT and Scaling kernel:* performs fused FHT and token scaling for Key, and applies FHT to Query. (ii) *Quantization kernel:* performs GPU-efficient quantization for both Key and Value. (iii) *Dequantization, De-Scaling, and Attention kernel:* handles dequantization for Key and Value, inverse scaling for Key, and attention computation.

The FHT is adopted for its computational efficiency over standard matrix multiplication, achieving $O(d \log d)$ complexity compared to $O(d^2)$, where d is the dimension (Ashkboos et al., 2024; Agarwal et al., 2024). *Omni-Token Scaling* leverages the hardware-accelerated `rsqrt` instruction, as motivated by our ablation study in Appendix U. Further implementation details of these CUDA kernels are provided in Appendix M, with storage accounting in Appendix M.6.

4 Experiments

4.1 Experimental Setup

Models and Tasks To comprehensively evaluate OScaR, we select three categories of LLMs: (i) text-only LLMs, including Llama-3.1-8B and Qwen3-8B (Grattafiori et al., 2024; Yang et al., 2025); (ii) multi-modal LLMs, including LLaVA-v1.6-vicuna-7B and Qwen3-VL-4B/8B-Instruct (Liu et al., 2023; 2024b; Bai et al., 2025); and (iii) omni-modal LLMs, including Qwen3-Omni-30B-A3B (Xu et al., 2025). These models represent a diverse set of open-source families and scales. To ensure a rigorous evaluation, most experiments focus on tasks requiring extreme long-context processing. Specifically, text-only LLMs are evaluated on LongBench-E and the Needle-in-a-Haystack (NIAH) benchmark (Bai et al., 2024; Kamradt, 2023); multi-modal LLMs on OCRBench and DocVQA (Liu

Table 1: LongBench-E evaluation results. All quantized methods except TurboQuant+ use INT2 quantization with a group size of 32, whereas TurboQuant+ uses 2.5-bit. TurboQuant is based on TurboQuant+ (Turney & Contributors, 2026); QJL is excluded as it degrades performance. Implementation details are in Appendix O.

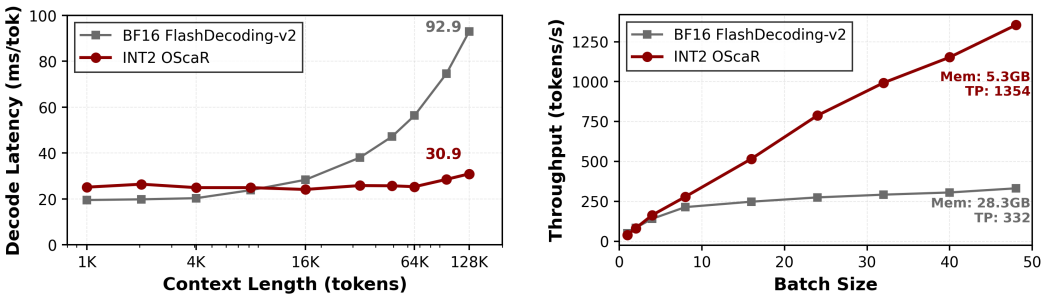
Model	Method	Hotpot QA	Qasper	Gov Report	Multi News	Trec	Trivia QA	Passage Retrieval	Repo Bench-P	Avg.
Llama-3.1-8B	16bit	13.76	11.86	29.88	5.89	72.67	90.15	43.57	65.78	41.70
	QuaRot	13.73	9.24	18.23	1.75	73.00	89.68	42.46	55.46	37.94
	RotateKV	13.13	10.97	17.82	1.28	72.33	**90.56**	38.83	58.95	37.98
	KIVI	**14.66**	10.71	24.41	2.70	73.00	89.83	38.95	64.42	39.84
	OTT	13.27	10.70	**29.45**	4.65	73.00	90.02	39.39	**65.45**	40.74
	TurboQuant+	13.85	10.04	26.60	3.18	72.33	89.91	41.51	62.82	40.03
	OScaR (ours)	13.68	**11.61**	**29.45**	**5.68**	**73.33**	**90.41**	**44.46**	65.39	**41.75**
Qwen3-8B	16bit	13.10	10.27	29.56	21.96	71.33	91.95	95.81	62.47	49.56
	QuaRot	11.70	7.47	20.13	15.01	33.00	89.52	91.33	52.88	40.13
	RotateKV	**12.74**	6.78	14.11	10.59	71.00	88.81	91.57	48.00	42.95
	KIVI	11.86	9.03	28.12	21.58	71.00	91.09	91.92	58.96	47.95
	OTT	11.86	9.26	**28.56**	**21.82**	71.00	91.36	92.39	59.41	48.21
	TurboQuant+	12.53	8.35	28.47	21.66	71.33	90.95	88.92	58.23	47.56
	OScaR (ours)	12.35	**9.94**	28.33	21.46	**72.00**	**91.46**	**92.78**	**61.57**	**48.74**

et al., 2024c; Mathew et al., 2021); and omni-modal LLMs on MMAU-Pro (Kumar et al., 2026). Detailed descriptions of these tasks are provided in Appendix N. Extensive ablation studies are provided in Appendix U, including analyses of the proposed innovations and alternative normalization strategies for *Omni-Token Scaling*. In addition, we provide visual comparisons of token norm distributions before and after applying OScaR in Appendix K. A Pareto front analysis of accuracy and efficiency is presented in Appendix V. A comprehensive efficiency evaluation is provided in Section 4.3 and Appendix W, assessing decoding speedup, memory savings, and throughput improvements.

Baselines We compare OScaR against several strong baselines, which fall into three categories: (i) per-channel Key quantization, including KIVI and OTT (Liu et al., 2024d; Su et al., 2025a); (ii) rotation-based per-token Key quantization, such as QuaRot and RotateKV (Ashkboos et al., 2024; Su et al., 2025c); and (iii) LUT-based methods, represented by TurboQuant (Zandieh et al., 2025a). Among these, OTT and RotateKV employ high-precision protection for outlier tokens. These baselines span diverse strategies, enabling a comprehensive evaluation. To ensure a fair comparison, we carefully align the configuration of each method. For QuaRot, we adopt only its KV cache quantization component. For KIVI, OTT, and OScaR, the residual length for per-channel quantization is uniformly set to 128. For OTT, the number of high-precision outlier tokens is set to 5. We also account for the average bit overhead introduced by quantization parameters. Specifically, TurboQuant employs 2.5-bit quantization, assigning higher bit-widths to outlier channels while using 2-bit for regular channels, whereas all other methods adopt INT2 quantization. Since TurboQuant does not provide an official code release, we use TurboQuant+ (Turney & Contributors, 2026), a widely adopted open-source implementation. Further implementation details of TurboQuant+ are provided in Appendix O.

4.2 Main Experimental Results

Results on Text-Only LLMs The LongBench-E results are presented in Table 1, and the NIAH experiment is shown in Figure 29 of Appendix P. On LongBench-E, OScaR achieves the highest average accuracy among all competing quantized methods, outperforming the second-best method by 1.01 percentage points (41.75% vs. 40.74%). Compared to the 16-bit baseline, OScaR incurs only a negligible accuracy drop of 1.7% on Qwen3-8B. In the NIAH task, OScaR achieves 96.5% retrieval



(a) Decoding latency across context lengths.

(b) Decoding throughput across batch sizes.

Figure 6: Efficiency analysis of OScaR against `BF16 FlashDecoding-v2`. Annotations show OScaR at 128K context length (latency) and batch size 48 (throughput and memory).

accuracy, significantly exceeding the second-best method (92.7%) and slightly surpassing the 16-bit baseline (96.0%), demonstrating its robustness in long-context retrieval scenarios.

Results on Multi-Modal and Omni-Modal LLMs Table 9 reports the OCRBench results, Table 10 summarizes the DocVQA results, and Table 11 presents the MMAU-Pro results for omni-modal LLMs. Across all benchmarks, OScaR maintains strong model performance under 2-bit quantization, frequently approaching the 16-bit baseline. On OCRBench, OScaR achieves a 0.7 percentage point improvement over the second-best method on Qwen3-VL-4B. On MMAU-Pro, it attains the highest scores among quantized methods across open-ended QA, Good Rate, and audio instruction following, surpassing the next-best method by 0.8, 2.8, and 4.6 percentage points, respectively. These results indicate that OScaR effectively preserves model capabilities in both multi-modal and omni-modal contexts. Further analyses appear in Appendices Q, R, and S.

4.3 Efficiency Analysis

In this section, we evaluate the efficiency of OScaR. Experiments are conducted on a single H20 GPU (141GB) using Qwen3-8B, with `BF16 FlashDecoding-V2` as the baseline. The evaluation consists of two parts: (i) measuring decoding latency under varying context lengths in a single-batch setting, and (ii) fixing the context length at 4K while increasing batch size to assess memory savings and corresponding throughput improvements.

As illustrated in Figure 6, OScaR achieves substantial efficiency gains, reflecting both its low computational complexity and the advantages of our system-level design. Specifically, at a context length of 128K, OScaR attains up to a 3.0× decoding speedup over the baseline. With a batch size of 48, it reduces the decoding memory footprint by 5.3× and increases throughput by 4.1×. Additional decoding efficiency comparisons with TurboQuant+ and controlled latency measurements are provided in Appendices W and W.1, respectively.

5 Conclusion

In this work, we revisited the fundamental limitations of per-channel KV cache quantization under extreme compression and identified TNI as a primary structural bottleneck that systematically amplifies quantization error. Motivated by this insight, we proposed *OScaR*, a lightweight and training-free KV cache compression framework for X-LLMs. By integrating *Canalized Rotation* with *Omni-Token Scaling*, OScaR effectively mitigates TNI-induced sequence-dimensional variance. We hope that OScaR can serve as a critical framework for efficient LLM inference and provide valuable guidance for KV cache quantization in LLMs and beyond.

AI USE STATEMENT

Generative AI tools were used solely for language editing, including grammar, spelling, word choice, and minor improvements to clarity and consistency. All technical content, experimental results, analyses, and conclusions were developed and verified by the authors. The authors take full responsibility for the final manuscript.

ETHICS STATEMENT

This work studies efficient inference through KV-cache compression of existing language and multimodal models. Compression does not remove the biases, privacy risks, or potential for harmful outputs inherited from those models. Use of the models, datasets, and third-party software remains subject to their applicable licenses and access conditions.

REPRODUCIBILITY STATEMENT

Section 4 and Appendix N describe the evaluation setup and benchmarks. Algorithm 1, Appendices M and H, and Appendices I.1 and W.1 provide algorithmic details, derivations, and additional measurement protocols. The anonymous supplementary archive includes packed-cache CUDA kernels, the Qwen3 system evaluation scripts, the available PyTorch fake-quantization implementation and LongBench scripts, and machine-readable tables for the additional measurements.

REFERENCES

Krish Agarwal, Rishi Astra, Adnan Hoque, Mudhakar Srivatsa, Raghu Ganti, Less Wright, and Sijia Chen. HadaCore: Tensor Core Accelerated Hadamard Transform Kernel. *arXiv preprint arXiv:2412.08832*, 2024.

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. Systematic Outliers in Large Language Models. *arXiv preprint arXiv:2502.06415*, 2025.

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, pp. 100213–100240, 2024. doi: 10.52202/079017-3180. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/b5b939436789f76f08b9d0da5e81af7c-Paper-Conference.pdf.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-VL Technical Report. *arXiv preprint arXiv:2511.21631*, 2025.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. LongBench: A Bilingual, Multitask Benchmark for Long Context Understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137, 2024. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172/.

Yelysei Bondarenko, Markus Nagel, and Tijmen Blankevoort. Quantizable Transformers: Removing Outliers by Helping Attention Heads Do Nothing. In *Advances in Neural Information Processing Systems*, volume 36, pp. 75067–75096, 2023. doi: 10.52202/075280-3282. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/edbcb7583fd8921dad78adecfe06a99b-Paper-Conference.pdf.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, et al. PyramidKV: Dynamic KV Cache Compression based on Pyramidal Information Funneling. *arXiv preprint arXiv:2406.02069*, 2024.

Ruisheng Cao, Mouxiang Chen, Jiawei Chen, Zeyu Cui, Yunlong Feng, Binyuan Hui, Yuheng Jing, Kaixin Li, Mingze Li, Junyang Lin, et al. Qwen3-Coder-Next Technical Report. *arXiv preprint arXiv:2603.00729*, 2026.

Dayou Du, Shijie Cao, Jianyi Cheng, Luo Mai, Ting Cao, and Mao Yang. BitDecoding: Unlocking Tensor Cores for Long-Context LLMs with Low-Bit KV Cache. *arXiv preprint arXiv:2503.18773*, 2025.

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. *arXiv preprint arXiv:2405.06219*, 2024.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers. *arXiv preprint arXiv:2210.17323*, 2022.

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. *arXiv preprint arXiv:2310.01801*, 2023.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*, 2024.

Tianyu Guo, Druv Pai, Yu Bai, Jiantao Jiao, Michael I. Jordan, and Song Mei. Active-Dormant Attention Heads: Mechanistically Demystifying Extreme-Token Phenomena in LLMs. *arXiv preprint arXiv:2410.13835*, 2024a.

Zhiyu Guo, Hidetaka Kamigaito, and Taro Watanabe. Attention Score is not All You Need for Token Importance Indicator in KV Cache Reduction: Value Also Matters. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 21158–21166, 2024b. doi: 10.18653/v1/2024.emnlp-main.1178. URL https://aclanthology.org/2024.emnlp-main.1178/.

Insu Han, Praneeth Kacham, Amin Karbasi, Vahab Mirrokni, and Amir Zandieh. PolarQuant: Quantizing KV Caches with Polar Transformation. *arXiv preprint arXiv:2502.02617*, 2025.

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*, volume 37, pp. 68287–68307, 2024. doi: 10.52202/079017-2181. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/7e57131fdeb815764434b65162c88895-Paper-Conference.pdf.

Jack Hong, Shilin Yan, Jiayin Cai, Xiaolong Jiang, Yao Hu, and Weidi Xie. WorldSense: Evaluating Real-world Omnimodal Understanding for Multimodal LLMs. *arXiv preprint arXiv:2502.04326*, 2025.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun S. Shao, Kurt Keutzer, and Amir Gholami. KVQuant: Towards 10 Million Context Length LLM Inference with KV Cache Quantization. In *Advances in Neural Information Processing Systems*, volume 37, pp. 1270–1303, 2024. doi: 10.52202/079017-0040. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/028fcbcf85435d39a40c4d61b42c99a4-Paper-Conference.pdf.

Zhongping Ji. IsoQuant: Hardware-Aligned SO(4) Isoclinic Rotations for LLM KV Cache Compression. *arXiv preprint arXiv:2603.28430*, 2026.

Mingyu Jin, Kai Mei, Wujiang Xu, Mingjie Sun, Ruixiang Tang, Mengnan Du, Zirui Liu, and Yongfeng Zhang. Massive Values in Self-Attention Modules are the Key to Contextual Knowledge Understanding. *arXiv preprint arXiv:2502.01563*, 2025.

Greg Kamradt. LLMTest_NeedleInAHaystack. GitHub, 2023. URL https://github.com/gkamradt/LLMTest_NeedleInAHaystack.

Seil Kang, Jinyeong Kim, Junhyeok Kim, and Seong Jae Hwang. See What You Are Told: Visual Attention Sink in Large Multimodal Models. In *International Conference on Learning Representations*, volume 2025, pp. 87676–87703, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/da8a39bc39ae1c89dd6ebb1e3bcbb3f3-Paper-Conference.pdf.

Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, et al. Kimi Linear: An Expressive, Efficient Attention Architecture. *arXiv preprint arXiv:2510.26692*, 2025.

Sonal Kumar, Šimon Sedláček, Vaibhavi Lokegaonkar, Fernando López, Wenyi Yu, Nishit Anand, Hyeonggon Ryu, Lichang Chen, Maxim Plička, Miroslav Hlaváček, et al. MMAU-Pro: A Challenging and Comprehensive Benchmark for Holistic Evaluation of Audio General Intelligence. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 22688–22697, 2026. doi: 10.1609/aaai.v40i27.39430. URL https://ojs.aaai.org/index.php/AAAI/article/view/39430.

Haoyang Li, Yiming Li, Anxin Tian, Tianhao Tang, Zhanchao Xu, Xuejia Chen, Nicole Hu, Wei Dong, Qing Li, and Lei Chen. A Survey on Large Language Model Acceleration based on KV Cache Management. *Transactions on Machine Learning Research*, 2025a. URL https://openreview.net/forum?id=z3JZzu9EA3.

Kunjun Li, Zigeng Chen, Cheng-Yen Yang, and Jenq-Neng Hwang. Memory-Efficient Visual Autoregressive Modeling with Scale-Aware KV Cache Compression. *arXiv preprint arXiv:2505.19602*, 2025b.

Guang Liang, Jie Shao, Ningyuan Tang, Xinyao Liu, and Jianxin Wu. TWEO: Transformers Without Extreme Outliers Enables FP8 Training And Quantization For Dummies. *arXiv preprint arXiv:2511.23225*, 2025.

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*, volume 6, pp. 87–100, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/file/42a452cbafa9dd64e9ba4aa95cc1ef21-Paper-Conference.pdf.

Yujun Lin, Haotian Tang, Shang Yang, Zhekai Zhang, Guangxuan Xiao, Chuang Gan, and Song Han. QServe: W4A8KV4 Quantization and System Co-design for Efficient LLM Serving. In *Proceedings of Machine Learning and Systems*, volume 7, 2025. URL https://proceedings.mlsys.org/paper_files/paper/2025/file/fbe2b2f74a2ece8070d8fb073717bda6-Paper-Conference.pdf.

Akide Liu, Jing Liu, Zizheng Pan, Yefei He, Gholamreza Haffari, and Bohan Zhuang. MiniCache: KV Cache Compression in Depth Dimension for Large Language Models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 139997–140031, 2024a. doi: 10.52202/079017-4443. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/fd0705710bf01b88a60a3d479ea341d9-Paper-Conference.pdf.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual Instruction Tuning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 34892–34916, 2023. doi: 10.52202/075280-1516. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/6dcf277ea32ce3288914faf369fe6de0-Paper-Conference.pdf.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved Baselines with Visual Instruction Tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 26296–26306, 2024b. URL https://openaccess.thecvf.com/content/CVPR2024/html/Liu_Improved_Baselines_with_Visual_Instruction_Tuning_CVPR_2024_paper.html.

Yanyu Liu, Jingying Fu, Sixiang Liu, Yitian Zou, Shouhua Zhang, and Jiehan Zhou. KV Cache Compression for Inference Efficiency in LLMs: A Review. In *Proceedings of the 4th International Conference on Artificial Intelligence and Intelligent Information Processing*, pp. 207–212, 2025. doi: 10.1145/3778534.3778567.

Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xu-Cheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. OCRBench: on the hidden mystery of OCR in large multimodal models. *Science China Information Sciences*, 67(12):220102, 2024c. doi: 10.1007/s11432-024-4235-6.

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. *arXiv preprint arXiv:2402.02750*, 2024d.

Minesh Mathew, Dimosthenis Karatzas, and C.V. Jawahar. DocVQA: A Dataset for VQA on Document Images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 2200–2209, 2021. URL https://openaccess.thecvf.com/content/WACV2021/html/Mathew_DocVQA_A_Dataset_for_VQA_on_Document_Images_WACV_2021_paper.html.

Meituan LongCat Team, Bayan, Bei Li, Bingye Lei, Bo Wang, Bolin Rong, Chao Wang, Chao Zhang, Chen Gao, Chen Zhang, et al. LongCat-Flash Technical Report. *arXiv preprint arXiv:2509.01322*, 2025a.

Meituan LongCat Team, Xunliang Cai, Qilong Huang, Zhuoliang Kang, Hongyu Li, Shijun Liang, Liya Ma, Siyu Ren, Xiaoming Wei, Rixu Xie, et al. LongCat-Video Technical Report. *arXiv preprint arXiv:2510.22200*, 2025b.

Meituan LongCat Team, Anchun Gui, Bei Li, Bingyang Tao, Bole Zhou, Borun Chen, Chao Zhang, Chao Zhang, Chengcheng Han, Chenhui Yang, et al. Introducing LongCat-Flash-Thinking: A Technical Report. *arXiv preprint arXiv:2509.18883*, 2025c.

Meituan LongCat Team, Hanghang Ma, Haoxian Tan, Jiale Huang, Junqiang Wu, Jun-Yan He, Lishuai Gao, Songlin Xiao, Xiaoming Wei, Xiaoqi Ma, et al. LongCat-Image Technical Report. *arXiv preprint arXiv:2512.07584*, 2025d.

Meituan LongCat Team, Bairui Wang, Bayan, Bin Xiao, Bo Zhang, Bolin Rong, Borun Chen, Chang Wan, Chao Zhang, Chen Huang, et al. LongCat-Flash-Omni Technical Report. *arXiv preprint arXiv:2511.00279*, 2025e.

Meituan LongCat Team, Anchun Gui, Bei Li, Bingyang Tao, Bole Zhou, Borun Chen, Chao Zhang, Chao Zhang, Chen Gao, Chen Zhang, et al. LongCat-Flash-Thinking-2601 Technical Report. *arXiv preprint arXiv:2601.16725*, 2026a.

Meituan LongCat Team, Bin Xiao, Chao Wang, Chengjiang Li, Chi Zhang, Chong Peng, Hang Yu, Hao Yang, Haonan Yan, Haoze Sun, et al. LongCat-Next: Lexicalizing Modalities as Discrete Tokens. *arXiv preprint arXiv:2603.27538*, 2026b.

Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart van Baalen, and Tijmen Blankevoort. A White Paper on Neural Network Quantization. *arXiv preprint arXiv:2106.08295*, 2021.

OpenAI. gpt-oss-120b & gpt-oss-20b Model Card. *arXiv preprint arXiv:2508.10925*, 2025.

John D Pope. RotorQuant: Clifford Algebra Vector Quantization for LLM KV Cache Compression. Technical report, 2026. URL https://www.scrya.com/rotorquant.pdf.

Ziran Qin, Youru Lv, Mingbao Lin, Hang Guo, Zeren Zhang, Danping Zou, and Weiyao Lin. Head-Aware KV Cache Compression for Efficient Visual Autoregressive Modeling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 24982–24990, 2026. doi: 10.1609/aaai.v40i30.39686. URL https://ojs.aaai.org/index.php/AAAI/article/view/39686.

Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, et al. Gated Attention for Large Language Models: Non-linearity, Sparsity, and Attention-Sink-Free. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 100092–100118, 2025. doi: 10.52202/085713-3345. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/904e89bb4e632e75fb47f093b620b257-Paper-Conference.pdf.

Yi Su, Yuechi Zhou, Quantong Qiu, Juntao Li, Qingrong Xia, Ping Li, Xinyu Duan, Zhefeng Wang, and Min Zhang. Accurate KV Cache Quantization with Outlier Tokens Tracing. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12895–12915, 2025a. doi: 10.18653/v1/2025.acl-long.631. URL https://aclanthology.org/2025.acl-long.631/.

Zunhai Su and Kehong Yuan. KVSink: Understanding and Enhancing the Preservation of Attention Sinks in KV Cache Quantization for LLMs. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=gIqb6zWZoO.

Zunhai Su, Wang Shen, Linge Li, Zhe Chen, Hanyu Wei, Huangqi Yu, and Kehong Yuan. AKVQ-VL: Attention-Aware KV Cache Adaptive 2-Bit Quantization for Vision-Language Models. In *2025 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, 2025b. doi: 10.1109/icme59968.2025.11209367.

Zunhai Su, Hanyu Wei, Zhe Chen, Wang Shen, Linge Li, Huangqi Yu, and Kehong Yuan. RotateKV: Accurate and Robust 2-Bit KV Cache Quantization for LLMs via Outlier-Aware Adaptive Rotations. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, pp. 6200–6208, 2025c. doi: 10.24963/ijcai.2025/690. URL https://doi.org/10.24963/ijcai.2025/690.

Zunhai Su, Qingyuan Li, Hao Zhang, Weihao Ye, Qibo Xue, Yulei Qian, Ngai Wong, and Kehong Yuan. Unveiling Super Experts in Mixture-of-Experts Large Language Models. In *International Conference on Learning Representations*, volume 2026, pp. 118323–118354, 2026a. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/c01b90b643d0e877cff0f34cbe27410f-Paper-Conference.pdf.

Zunhai Su, Weihao Ye, Hansen Feng, Keyu Fan, Jing Zhang, Dahai Yu, Zhengwu Liu, and Ngai Wong. XStreamVGGT: Extremely Memory-Efficient Streaming Vision Geometry Grounded Transformer With KV Cache Compression. *Journal of the Society for Information Display*, 34(5):535–545, 2026b. doi: 10.1002/jsid.70086.

Zunhai Su, Hengyuan Zhang, Wei Wu, Yifan Zhang, Yaxiu Liu, He Xiao, Qingyao Yang, Yuxuan Sun, Rui Yang, Chao Zhang, et al. Attention Sink in Transformers: A Survey on Utilization, Interpretation, and Mitigation. *arXiv preprint arXiv:2604.10098*, 2026c.

Mingjie Sun, Xinlei Chen, J. Zico Kolter, and Zhuang Liu. Massive Activations in Large Language Models. *arXiv preprint arXiv:2402.17762*, 2024.

Keda Tao, Haoxuan You, Yang Sui, Can Qin, and Huan Wang. Plug-and-Play 1.x-Bit KV Cache Quantization for Video Large Language Models. *arXiv preprint arXiv:2503.16257*, 2025.

Chongjun Tu, Peng Ye, Dongzhan Zhou, Lei Bai, Gang Yu, Tao Chen, and Wanli Ouyang. Attention Reallocation: Towards Zero-cost and Controllable Hallucination Mitigation of MLLMs. *International Journal of Computer Vision*, 134(1):22, 2026. doi: 10.1007/s11263-025-02607-z.

Tom Turney and Contributors. TurboQuant+. GitHub repository, 2026. URL https://github.com/TheTom/turboquant_plus. Accessed 2026-09-25.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Zhongwei Wan, Ziang Wu, Che Liu, Jinfa Huang, Zhihong Zhu, Peng Jin, Longyue Wang, and Li Yuan. LOOK-M: Look-Once Optimization in KV Cache for Efficient Multimodal Long-Context Inference. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 4065–4078, 2024. doi: 10.18653/v1/2024.findings-emnlp.235. URL https://aclanthology.org/2024.findings-emnlp.235/.

Jianing Wang, Jianfei Zhang, Qi Guo, Linsen Guo, Rumei Li, Chao Zhang, Chong Peng, Cunguang Wang, Dengchang Zhao, Jiarong Shi, et al. LongCat-Flash-Prover: Advancing Native Formal Reasoning via Agentic Tool-Integrated Reinforcement Learning. *arXiv preprint arXiv:2603.21065*, 2026.

Xiuying Wei, Yunchen Zhang, Yuhang Li, Xiangguo Zhang, Ruihao Gong, Jinyang Guo, and Xianglong Liu. Outlier Suppression+: Accurate quantization of large language models by equivalent and effective shifting and scaling. In *Proceedings of the 2023 Conference on Empirical Methods in*

Natural Language Processing, pp. 1648–1665, 2023. doi: 10.18653/v1/2023.emnlp-main.102. URL https://aclanthology.org/2023.emnlp-main.102/.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dLLM: Training-free Acceleration of Diffusion LLM by Enabling KV Cache and Parallel Decoding. *arXiv preprint arXiv:2505.22618*, 2025.

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 *International Conference on Machine Learning*, pp. 38087–38099. PMLR, 2023a. URL https://proceedings.mlr.press/v202/xiao23c.html.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient Streaming Language Models with Attention Sinks. *arXiv preprint arXiv:2309.17453*, 2023b.

He Xiao, Qingyao Yang, Dirui Xie, Wendong XU, Zunhai Su, Runming Yang, Haobo Liu, Wenyong Zhou, Zhengwu Liu, and Ngai Wong. Exploring Layer-wise Information Effectiveness for Post-Training Quantization in Small Language Models. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 15753–15764, 2026. doi: 10.18653/v1/2026.findings-acl.771. URL https://aclanthology.org/2026.findings-acl.771/.

Jing Xiong, Liyang Fan, Hui Shen, Zunhai Su, Min Yang, Lingpeng Kong, and Ngai Wong. DoPE: Denoising Rotary Position Embedding. *arXiv preprint arXiv:2511.09146*, 2025.

Jin Xu, Zhifang Guo, Hangrui Hu, Yunfei Chu, Xiong Wang, Jinzheng He, Yuxuan Wang, Xian Shi, Ting He, Xinfa Zhu, et al. Qwen3-Omni Technical Report. *arXiv preprint arXiv:2509.17765*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025.

Amir Zandieh, Majid Daliri, Majid Hadian, and Vahab Mirrokni. TurboQuant: Online Vector Quantization with Near-optimal Distortion Rate. *arXiv preprint arXiv:2504.19874*, 2025a.

Amir Zandieh, Majid Daliri, and Insu Han. QJL: 1-Bit Quantized JL Transform for KV Cache Quantization with Zero Overhead. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 25805–25813, 2025b. doi: 10.1609/aaai.v39i24.34773. URL https://ojs.aaai.org/index.php/AAAI/article/view/34773.

Hengyuan Zhang, Xinrong Chen, Zunhai Su, Xiao Liang, Jing Xiong, Wendong Xu, He Xiao, Chaofan Tao, Wei Zhang, Ruobing Xie, et al. Beyond Outliers: A Data-Free Layer-wise Mixed-Precision Quantization Approach Driven by Numerical and Structural Dual-Sensitivity. *arXiv preprint arXiv:2603.17354*, 2026a.

Hengyuan Zhang, Zhihao Zhang, Mingyang Wang, Zunhai Su, Yiwei Wang, Qianli Wang, Shuzhou Yuan, Ercong Nie, Xufeng Duan, Qibo Xue, et al. Locate, steer, and improve: A practical survey of actionable mechanistic interpretability in large language models. *Computer Science Review*, 62: 101011, 2026b. doi: 10.1016/j.cosrev.2026.101011.

Dong Zhuo, Wenzhao Zheng, Jiahe Guo, Yuqi Wu, Jie Zhou, and Jiwen Lu. Streaming 4D Visual Geometry Transformer. *arXiv preprint arXiv:2507.11539*, 2025.

Zayd M. K. Zuhri, Erland Hilman Fuadi, and Alham Fikri Aji. Softpick: No Attention Sink, No Massive Activations with Rectified Softmax. *arXiv preprint arXiv:2504.20966*, 2025.

APPENDIX CONTENTS

A Limitations and Future Directions **18**

B Related Work **18**

B.1 KV Cache Quantization 18

B.2 Outliers in Large Language Models 18

C Algorithm of OScaR **19**

D Preliminaries on Low-Bit Quantization **19**

E Token Norm Imbalance in Text-Only LLMs **19**

F Outlier Tokens and Attention Sinks **20**

G Token Norm Imbalance in Multi-modal LLMs **21**

H Theoretical Derivation of TNI-Induced Quantization Errors **21**

H.1 Connection to attention-logit error 22

I Quantitative Analysis of TNI-Induced Quantization Errors **22**

I.1 Real-tensor calibration of the error model 23

J Detailed Analysis of Scaling-Induced Outlier Artifact **23**

J.1 Comparison with rotation and selective token protection 24

K Additional Visualizations of OScaR Processing Stages **24**

L Theoretical Complexity Analysis of KV Cache Quantization Methods **24**

L.1 Symbolic Operation Counts 25

L.2 Effective Cost Conversion 26

L.3 Numerical Results 26

L.4 Discussion 26

M Implementation Details of OScaR's CUDA Kernels **26**

M.1 Overall Design 27

M.2 Fused Hadamard-Norm Kernel 27

M.3 Quantization Format and Cache Organization 27

M.4 Prefill and Decode Workflows 28

M.5 Summary of Key Differences from the Baseline 28

M.6 Packed-cache storage and norm metadata 28

N Details of Datasets and Benchmarks **29**

N.1 Text-Only LLM Benchmarks 29

N.2 Multi-modal LLM Benchmarks 29

N.3 Omni-modal LLM Benchmark 30

O Additional TurboQuant+ Implementation Details **30**

P Experimental Results and Analysis on Needle-in-a-Haystack **31**

Q Experimental Results and Analysis on OCRBench **31**

R Experimental Results and Analysis on DocVQA **32**

S Experimental Results and Analysis on MMAU-Pro **32**

T TNI Analysis Before and After OScaR **33**

U Ablation Study **33**

V Accuracy-Efficiency Pareto Front Analysis **34**

W Additional Decoding Efficiency Measurements **34**

W.1 Controlled end-to-end and component-wise latency 35

A Limitations and Future Directions

Although OScaR imposes lower overhead than most existing quantization frameworks, its online rotation and token-wise scaling operations incur non-trivial computational costs relative to plain per-channel quantization. Specifically, *Canalized Rotation* requires online computation in the presence of RoPE, which precludes offline fusion of Query and Key Hadamard transforms with weight matrices (Ashkboos et al., 2024; Su et al., 2025c). In our system design, we mitigate this overhead via two key strategies: (i) employing HadaCore (Agarwal et al., 2024) for efficient token-wise FHT computation on Tensor Cores; and (ii) fusing kernels to consolidate (FHT + scaling) and (dequantization + de-scaling + attention) into dedicated CUDA operators. Future work may explore alternative *Canalized Rotation* that further reduce online overhead or enable more efficient hardware-aware implementations.

Furthermore, OScaR represents a highly general framework for KV cache quantization, applicable to LLMs and beyond (i.e., multi-modal and omni-modal LLMs). However, our current experiments are primarily conducted on models with LLM backbones. We posit that OScaR can be applied to autoregressive inference beyond LLMs that requires KV caching, including streaming vision models such as StreamVGGT (Su et al., 2026b; Zhuo et al., 2025), visual autoregressive models (Li et al., 2025b; Qin et al., 2026), and diffusion LLMs with KV cache (Wu et al., 2025). However, these models often exhibit architectural characteristics that differ substantially from standard LLM backbones, and we have not yet conducted extensive experiments in these domains. Moreover, KV cache compression in such models remains an emerging area. We leave a more thorough evaluation across diverse model families to future work.

B Related Work

B.1 KV Cache Quantization

Quantization is essential for efficient deployment of LLMs, with seminal works such as GPTQ, AWQ, and SmoothQuant establishing effective methods for weight and activation compression (Frantar et al., 2022; Lin et al., 2024; Xiao et al., 2023a; 2026; Zhang et al., 2026a). As context lengths increase, the KV cache has emerged as the dominant memory bottleneck during decoding, necessitating specialized quantization strategies (Liu et al., 2024d; Li et al., 2025a). Existing approaches can be broadly categorized by their quantization granularity: per-token, per-channel, and per-element paradigms. Per-token quantization aligns with the incremental dynamics of auto-regressive decoding but remains vulnerable to persistent channel-wise outliers in Key tensors (Liu et al., 2024d; Hooper et al., 2024). To address this, methods such as QuaRot, RotateKV, and ZipCache employ transformations including rotation and smoothing to redistribute outlier energy (Ashkboos et al., 2024; Su et al., 2025c; He et al., 2024; Duanmu et al., 2024). Per-channel approaches, including KIVI, KVQuant, and OTT, exploit intrinsic channel-wise outlier distributions to reduce quantization difficulty (Liu et al., 2024d; Hooper et al., 2024; Su et al., 2025a). Recently, per-element paradigms such as TurboQuant and its extensions leverage randomized rotations combined with residual error correction to achieve KV cache compression (Zandieh et al., 2025a; Pope, 2026; Ji, 2026; Zandieh et al., 2025b; Han et al., 2025). While these methods provide rigorous theoretical guarantees, their complex pipelines often result in high implementation overhead and practical deviations during deployment. Despite these advancements, accurate and lightweight KV cache compression at extreme bit-widths remains a challenging problem. Moreover, specialized studies on multi-modal and omni-modal LLMs are still limited.

B.2 Outliers in Large Language Models

Outliers in LLMs fundamentally disrupt numerical precision and pose a critical challenge for high-fidelity quantization (Nagel et al., 2021; Wei et al., 2023; Sun et al., 2024; Su & Yuan, 2025; Su et al., 2026c; Zhang et al., 2026b). These outliers can be broadly categorized as channel-wise and token-wise based on their distributional characteristics. Channel-wise outliers exhibit disproportionately large magnitudes in specific feature dimensions, predominantly appearing in Key and Query tensors while remaining comparatively subdued in Value tensors (Liu et al., 2024d; Hooper et al., 2024; Jin et al., 2025). Token-level outliers manifest in two distinct forms. The first consists of systematic

activation outlier tokens arising from the outputs of down-projection layers and inter-block hidden states, which can reach magnitudes tens of thousands of times larger than the median, severely destabilizing activation quantization (Sun et al., 2024; Su et al., 2026a; Ashkboos et al., 2024; An et al., 2025). The second consists of attention outlier tokens, where specific tokens exhibit markedly reduced norms across Query, Key, and Value tensors (Su & Yuan, 2025; Bondarenko et al., 2023; Guo et al., 2024b;a). Both channel-wise outliers and the second form of token-level outliers are closely associated with representational collapse under extreme KV cache compression. While per-channel paradigms and equivalent transformations can effectively mitigate channel-wise impacts (Xiao et al., 2023a; Ashkboos et al., 2024; Duanmu et al., 2024; Lin et al., 2025), existing methods often inadequately address token-level outliers. Techniques such as OTT and RotateKV trace and preserve a small number of outlier tokens with high precision in text-only LLMs, but they introduce hardware fragmentation and mixed-precision overheads, limiting the achievable effective compression (Su et al., 2025a;c; Su & Yuan, 2025; Duanmu et al., 2024; Su et al., 2025b; Hooper et al., 2024). In this work, we further characterize TNI across X-LLMs. OScaR addresses TNI through *Canalized Rotation* and *Omni-Token Scaling*, enabling uniform and efficient mitigation of TNI, including principled handling of outlier tokens.

C Algorithm of OScaR

This section presents the OScaR algorithm discussed in Section 3. As shown in Algorithm 1, we first describe the preprocessing step that applies Hadamard transforms to $\mathbf{W}_V$ and $\mathbf{W}_O$, followed by the pseudocode for OScaR during attention computation in both the prefill and decoding phases.

D Preliminaries on Low-Bit Quantization

Low-bit quantization is an effective method to reduce the memory footprint of the KV cache (Liu et al., 2024d; Li et al., 2025a; Liu et al., 2025). Consider the widely adopted asymmetric uniform quantization scheme, which maps floating-point values to b-bit integers. This scheme is parameterized by the quantization step size Δ and zero-point z, computed from the dynamic range $[x_{\min}, x_{\max}]$:

$$\Delta = \frac{x_{\max} - x_{\min}}{2^b - 1}, \quad z = \left\lfloor -\frac{x_{\min}}{\Delta} \right\rceil. \tag{6}$$

The step size Δ determines the numerical resolution, with larger values yielding coarser representations and higher quantization error (Nagel et al., 2021). The quantized integer $Q(x)$ and its reconstruction $\hat{x}$ are given by:

$$Q(x) = \text{clamp}\left(\left\lfloor \frac{x}{\Delta} \right\rceil + z, 0, 2^b - 1\right), \quad \hat{x} = (Q(x) - z) \cdot \Delta. \tag{7}$$

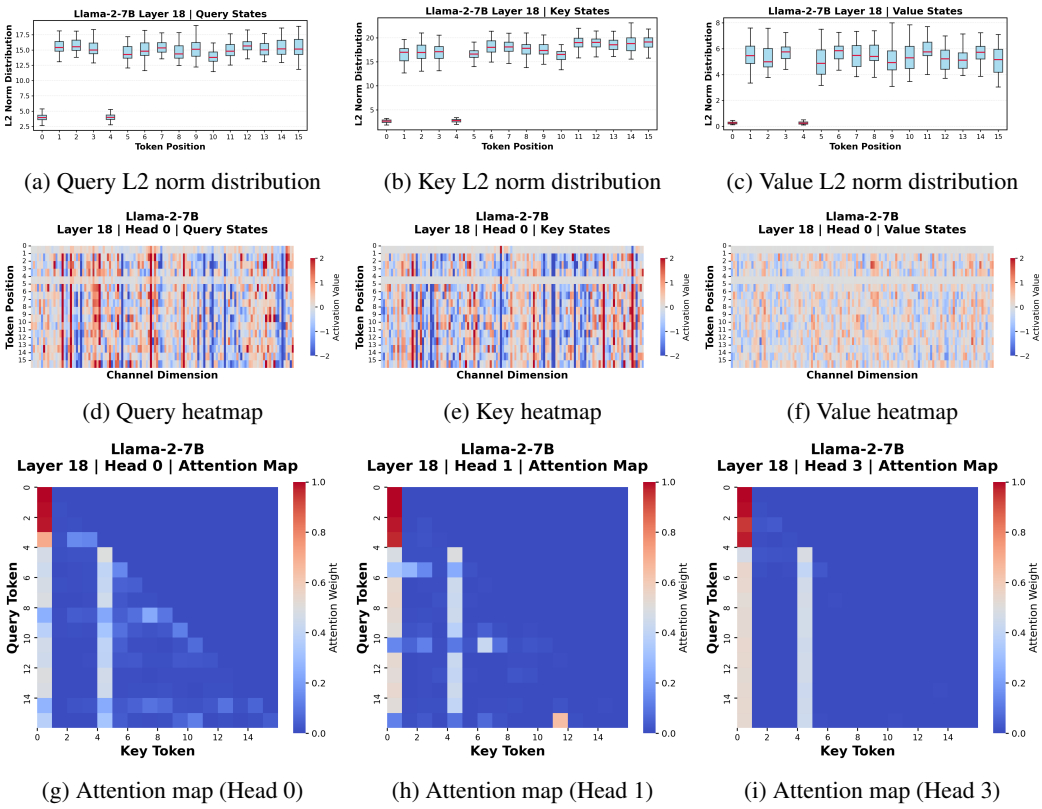
For extreme low-bit settings (e.g., $b \leq 4$), the reduced numerical resolution often leads to severe accuracy degradation, particularly in the presence of outliers. This necessitates outlier-aware strategies, including rotation-based energy redistribution (Ashkboos et al., 2024; Su et al., 2025c), residual error correction (Zandieh et al., 2025a), or mixed-precision preservation of salient tokens (Liu et al., 2024d; Su et al., 2025a).

E Token Norm Imbalance in Text-Only LLMs

As discussed in Section 3.1, we visualize the L2 norm distributions and heatmaps of Query, Key, and Value states across multiple text-only LLMs using the following prompt:

```
Summer is warm.\nWinter is cold.\nSpring is mild.\nAutumn is
crisp.
```

The results are presented in Figures 10, 11, 12, 13, 14, 15, 16, 17, and 18. Across all cases, low-norm outlier tokens are clearly observed in Q, K, and V states. These tokens are sparse yet consistent across attention states. Under per-channel quantization, these outliers inflate the per-channel scaling factors, leading to substantial quantization error, resulting in substantial fidelity degradation.



(a) Query L2 norm distribution (b) Key L2 norm distribution (c) Value L2 norm distribution

(d) Query heatmap (e) Key heatmap (f) Value heatmap

(g) Attention map (Head 0) (h) Attention map (Head 1) (i) Attention map (Head 3)

Figure 7: L2 norm distributions (row 1), value heatmaps (row 2), and attention maps (row 3) of Query, Key, and Value states. Each attention state contains a sparse yet consistent subset of tokens with exceptionally low norms. These low-norm outlier tokens correspond exactly to the same token positions as Attention Sink tokens.

F Outlier Tokens and Attention Sinks

As shown in Figure 7, outlier tokens remain consistent not only across Query, Key, and Value states but also with Attention Sinks, corroborating prior studies (Su et al., 2026c; Su & Yuan, 2025). A widely accepted explanation for this behavior is the *softmax limitation and no-op theory* (Bondarenko et al., 2023), which we briefly recapitulate below.

In standard attention, the sum-to-one softmax constraint requires that, for each query, the attention weights over all keys normalize to unity. When a query does not meaningfully align with any key in the context, the mechanism lacks a natural "null" option and is therefore forced to distribute attention mass to uninformative tokens. Consequently, attention heads learn to circumvent this constraint by adopting a no-op behavior. Let $\mathcal{S}$ denote the set of sink tokens (e.g., `[SEP]`, punctuation, or background patches). The resulting attention pattern can be approximated as:

$$A_{ij} \approx \begin{cases} 1, & j \in \mathcal{S} \\ 0, & \text{otherwise} \end{cases} \qquad \text{with} \qquad \|V_{\mathcal{S}}\| \approx 0, \tag{8}$$

where nearly all attention mass concentrates on sink tokens, whose Value vectors exhibit low norms, thereby producing minimal updates to the residual representation. This compression phenomenon also extends beyond Value states to Query and Key states (Su & Yuan, 2025).

Studies demonstrate that Attention-Sink-related extreme tokens can compromise training stability and hinder low-precision deployment (Qiu et al., 2025; Liang et al., 2025; Su et al., 2026a). Moreover, misallocated attention to uninformative tokens inherently limits overall model capacity (Kang et al., 2025; Tu et al., 2026). Consequently, developing robust mitigation frameworks has emerged as a critical research frontier. Recent efforts have focused on systematically explaining and eliminating

Attention Sinks, proposing approaches such as gated attention, modified softmax, and explicit attention bias (Qiu et al., 2025; Zuhri et al., 2025; Xiong et al., 2025; OpenAI, 2025).

G Token Norm Imbalance in Multi-modal LLMs



Figure 8: Example image used as visual input.

As discussed in Section 3.1, we visualize the L2 norm distributions of Query, Key, and Value states in multi-modal LLMs. The input is formatted using the model's chat template with `add_generation_prompt=True`, resulting in the token sequence shown below, where `</td>` denotes the sequence of image patch tokens corresponding to the example image in Figure 8:

```
<|im_start|>user
<|vision_start|></td><|vision_end|>What is in this image?<|im_end|>
<|im_start|>assistant
```

Representative results are presented in Figures 19, 20, and 21, which respectively demonstrate three patterns of TNI in multi-modal LLMs beyond attention-sink-related low-norm tokens: (i) broader token norm variation relative to text-only LLMs (Figure 19); (ii) inter-modality norm disparities, wherein norms remain smooth within each modality yet diverge substantially across modalities (Figure 20); and (iii) exceptionally large-norm outlier tokens, which contrast with the low-norm attention sink tokens (Figure 21).

H Theoretical Derivation of TNI-Induced Quantization Errors

Building on the empirical observations of TNI across X-LLMs in Section 3.1, we now present a theoretical analysis of TNI-induced errors in per-channel quantization.

For a uniform b-bit quantizer, the quantization error ϵ is modeled as a random variable uniformly distributed over $[-\Delta_{j,g}/2, \Delta_{j,g}/2]$. The mean squared error (MSE) for the j-th channel in block g is

$$\mathrm{MSE}_{j,g} = \mathbb{E}[\epsilon^2] = \frac{1}{\Delta_{j,g}} \int_{-\Delta_{j,g}/2}^{\Delta_{j,g}/2} \epsilon^2 \, d\epsilon = \frac{\Delta_{j,g}^2}{12}. \tag{9}$$

The quantization step size $\Delta_{j,g}$ is determined by the sample range $\mathcal{R}_{j,g} = \max_{t \in g} K_{t,j} - \min_{t \in g} K_{t,j}$ and the bit width, i.e., $\Delta_{j,g} = \mathcal{R}_{j,g}/(2^b - 1)$. For any two tokens u, v in the same block g, the range satisfies $\mathcal{R}_{j,g} \geq |K_{u,j} - K_{v,j}|$. Substituting this into Eq. (9) yields a pairwise lower bound:

$$\mathrm{MSE}_{j,g} \geq \frac{(K_{u,j} - K_{v,j})^2}{12(2^b - 1)^2}. \tag{10}$$

Within block g, let $m = \arg\max_{t \in g} \|\mathbf{k}_t\|_2$ and $n = \arg\min_{t \in g} \|\mathbf{k}_t\|_2$ denote the tokens with the largest and smallest ℓ_2 norms, respectively. Applying Eq. (10) and summing over all channels gives a

conservative lower bound on the average reconstruction error:

$$\begin{aligned}\overline{\mathrm{MSE}}_g \triangleq \frac{1}{|g|}\sum_{t\in g}\mathrm{MSE}_t \;&\gtrsim\; \frac{1}{12(2^b-1)^2}\sum_{j=1}^{d}(K_{m,j}-K_{n,j})^2 \\ &= \frac{\|\mathbf{k}_m-\mathbf{k}_n\|_2^2}{12(2^b-1)^2} \geq \frac{\left(\|\mathbf{k}_m\|_2-\|\mathbf{k}_n\|_2\right)^2}{12(2^b-1)^2}.\end{aligned} \tag{11}$$

where $|g|$ denotes the block size and $\mathrm{MSE}_t = \sum_{j=1}^{d}\mathrm{MSE}_{j,g,t}$ represents the total quantization error across all channels for token t.

This inequality reveals that the reconstruction error of a per-channel quantization block is fundamentally governed by the range of token norms within the block. Thus, TNI-induced error amplification constitutes an intrinsic vulnerability of per-channel block-wise quantization.

Under the uniform-noise model, this bound connects per-channel reconstruction error to token norm disparity. The real-tensor calibration in Appendix I.1 evaluates the approximation empirically; the bound is not a universal guarantee for deterministic low-bit quantization.

H.1 Connection to attention-logit error

For a Key vector $\mathbf{k}$, let $\mathbf{e} = \widehat{\mathbf{k}} - \mathbf{k}$. The attention-logit perturbation for query $\mathbf{q}$ is exactly

$$\delta s = \frac{\mathbf{q}^\top \mathbf{e}}{\sqrt{d}}. \tag{12}$$

Under the zero-mean uniform-noise approximation and uncorrelated channel errors, the conditional logit MSE is

$$\mathbb{E}[\delta s^2 \mid \mathbf{q}] = \frac{1}{12d}\sum_j q_j^2 \Delta_{j,g}^2. \tag{13}$$

Thus, expansion of the shared range enters the attention-logit error through the quantization step sizes, weighted by the query. This relation connects the shared quantization range to attention-logit error under the stated noise assumptions. Downstream task accuracy also depends on attention patterns and nonlinear propagation; Table 12 evaluates the coupled CR/OTS design through task accuracy.

I Quantitative Analysis of TNI-Induced Quantization Errors

Bits	w/ OTs vs. w/o OTs				Mixed-Modality vs. Single-Modality									
	Per-Channel K		Per-Token V		Per-Channel K					Per-Token V				
	w/ OTs	w/o OTs	w/ OTs	w/o OTs	Mixed			Single		Mixed			Single	
					overall	text	vision	text	vision	overall	text	vision	text	vision
4	0.23	0.16	0.02	0.05	0.23	0.24	0.24	0.29	0.10	0.02	0.03	0.02	0.03	0.02
3	1.34	0.78	0.11	0.14	1.34	1.50	1.28	1.62	0.62	0.11	0.15	0.09	0.15	0.09
2	5.92	3.52	0.52	0.59	5.92	5.98	5.87	6.17	2.45	0.52	0.64	0.40	0.65	0.41

Table 2: Quantization error analysis. The left section compares quantization errors between groups with and without outlier tokens (OTs), while the right section compares errors between mixed-modality and single-modality groups. All values are scaled by a factor of 100.

In Section 3.1 and Appendix H, we presented empirical observations and theoretical analysis of TNI and its impact. Here, we conduct an empirical quantization error analysis under extreme KV cache compression to quantify these effects. Our analysis centers on two TNI patterns:

- **Impact of outlier tokens:** comparing quantization errors between groups that include low-norm outlier tokens and those from which these outliers have been removed.
- **Mixed-modality versus single-modality discrepancies:** comparing quantization errors between groups containing tokens from multiple modalities and those containing tokens from a single modality.

We conduct experiments on LLaVA-v1.5-7B. All experiments employ round-to-nearest (RTN) quantization with a group size of 32, and errors are measured using MSE. The results are summarized in Table 2. Below, we analyze the impact of TNI on both per-channel and per-token quantization.

Impact on Per-Channel Key Quantization As shown in Table 2, the presence of outlier tokens significantly amplifies per-channel Key quantization errors. Under 2-bit quantization, the error increases by approximately 68% compared to groups from which these outliers are removed. Furthermore, mixed-modality groups exacerbate quantization errors relative to single-modality settings, leading to a 140% increase for the visual component under the same 2-bit configuration.

Impact on Per-Token Value Quantization Value states lack channel-wise outliers, making per-token quantization the standard choice. Although TNI persists, per-token quantization confines norm variations to individual tokens and prevents cross-token interference. Consequently, the error amplification caused by TNI in per-channel schemes does not manifest under per-token quantization.

The above analysis confirms that TNI fundamentally undermines per-channel quantization while having negligible impact on per-token quantization, providing strong empirical validation for our assumption and theoretical derivations.

I.1 Real-tensor calibration of the error model

The approximation $\mathrm{MSE}_{\mathrm{pred}}(j,g) = \Delta_{j,g}^2/12$ isolates the effect of a shared per-channel quantization range. We evaluate post-RoPE, pre-CR/OTS Keys from Qwen3-8B Layer 18 on 16 sequences of 4K tokens from the Paul Graham Essays corpus, with group size 32 and the asymmetric round-to-nearest rule used in the paper. Correlations are computed per sequence and reported as mean $\pm$ standard deviation.

Table 3: Predicted and measured Key reconstruction error on real tensors. Ratio denotes measured MSE divided by predicted MSE.

Bits	Predicted MSE	Measured MSE	Ratio	Pearson (log MSE)	Spearman
4	0.009105	0.009095	0.9989	0.9872 ± 0.0002	0.9792 ± 0.0004
3	0.041810	0.041709	0.9976	0.9872 ± 0.0002	0.9792 ± 0.0004
2	0.227634	0.226362	0.9944	0.9869 ± 0.0002	0.9785 ± 0.0004

The measured-to-predicted ratios in Table 3 correspond to aggregate gaps of 0.11%, 0.24%, and 0.56% at 4, 3, and 2 bits, respectively. The close errors and strong correlations support the shared-range approximation in this evaluated setting; they do not establish its validity across other models, layers, inputs, or quantizers.

J Detailed Analysis of Scaling-Induced Outlier Artifact

As discussed in Section 3.2, direct token-wise scaling, while effective in unifying token norms, introduces the *Scaling-Induced Outlier Artifact* under per-channel quantization. We analyze this phenomenon through both mathematical formulation and concrete examples.

Mathematical Formulation Consider a normal token $\mathbf{a} \in \mathbb{R}^d$ dominated by an outlier channel. Without loss of generality, assume channel d satisfies $a_d \gg a_j$ for all $j \neq d$. Let $\mathbf{b} \in \mathbb{R}^d$ be a low-norm token with $b_j \approx c$ (a small constant) for all j. Their ℓ_2 norms are $\|\mathbf{a}\|_2 \approx |a_d|$ and $\|\mathbf{b}\|_2 \approx \sqrt{d} \cdot c$. Scaling both tokens to a target norm N yields scaling factors $\alpha = N/\|\mathbf{b}\|_2$ and $\beta = N/\|\mathbf{a}\|_2$. The resulting values are $\mathbf{a}' = \beta\mathbf{a}$ and $\mathbf{b}' = \alpha\mathbf{b}$.

The artifact condition captures $\mathbf{b}'$ becoming an outlier in channels where $\mathbf{a}'$ is small:

$$\alpha c \gg \beta \max_{j \neq d} a_j. \tag{14}$$

Since $\alpha/\beta = \|\mathbf{a}\|_2/\|\mathbf{b}\|_2 \gg 1$ and c and $\max_{j\neq d} a_j$ are of similar magnitude, the inequality holds, creating artificial outliers.

Direct Scaling: Numerical Demonstration A concrete example illustrates the failure. Set $N = 1$, $\mathbf{a} = [1, 1, 1, 100]$, and $\mathbf{b} = [0.1, 0.1, 0.1, 0.1]$. Then $\|\mathbf{a}\|_2 \approx 100.015$, $\|\mathbf{b}\|_2 = 0.2$, giving $\beta \approx 0.01$ and $\alpha = 5$:

$$\mathbf{a}' \approx [0.01, 0.01, 0.01, 1.00], \qquad \mathbf{b}' = [0.5, 0.5, 0.5, 0.5]. \tag{15}$$

Observe that $\mathbf{b}'$ substantially exceeds $\mathbf{a}'$ in channels 1–3 (0.5 vs. 0.01). In per-channel quantization, this expands the dynamic range of these channels from approximately 0.01 to 0.5, increasing the quantization step by a factor of 50 and severely degrading precision for normal tokens.

J.1 Comparison with rotation and selective token protection

The distinction between balancing all token norms and protecting selected tokens is summarized in Table 4.

Table 4: Mechanistic distinction from representative alternatives. Selective high-precision token protection is distinct from the residual window used for online block quantization.

Method	Key granularity	Channel treatment	Treatment of token disparity
QuaRot (Ashkboos et al., 2024)	Per-token	Hadamard rotation	Separate token ranges; no all-token norm balancing
RotateKV (Su et al., 2025c)	Per-token	Outlier-aware rotation	Selective high-precision sink-token protection
OTT (Su et al., 2025a)	Per-channel	Channel-specific ranges	Tracing and high-precision protection of selected outlier tokens
OScaR	Per-channel	Canalized Rotation	Regular scaling of every Key token; no selective high-precision token pool

Table 4 separates quantization granularity from token treatment. Per-token ranges avoid sharing a range across tokens, but still share a range across channels within each token. Selective protection targets a subset of exceptional tokens. OScaR instead conditions the distribution of every Key token before per-channel quantization, covering both sparse outliers and broader inter-modality norm gaps with one regular dataflow.

K Additional Visualizations of OScaR Processing Stages

In this section, we provide additional visualizations of the OScaR processing stages. As shown in Figures 22, 23, and 24, direct scaling balances token norms but introduces the *Scaling-Induced Outlier Artifact*. *Canalized Rotation* alone fails to balance token norms. Only the complete OScaR framework successfully addresses TNI without incurring this artifact.

L Theoretical Complexity Analysis of KV Cache Quantization Methods

We analyze the theoretical computational overhead introduced by five KV cache quantization methods. Symbolic costs are per token during prefill and per step during decode; the numerical table multiplies prefill costs by the prompt length L. Each arithmetic operation (multiplication, addition, comparison, rounding, square root) is counted as one operation. Table lookups (LUT) are accounted for separately. The analysis excludes attention computation to isolate the overhead incurred by quantization and its auxiliary transformations. We focus specifically on key processing, as this constitutes the core distinguishing factor among the compared methods.

We emphasize that while every effort has been made to ensure a fair and theoretically sound analysis, the resulting estimates may deviate from actual hardware efficiency due to factors such as memory bandwidth, parallelism, kernel launch overhead, and operator fusion. Therefore, this theoretical analysis should be interpreted as providing comparative insight rather than precise performance predictions. A comprehensive empirical evaluation is presented in Section 4.3.

Table 5: Summary of raw operation counts (symbolic expressions).

Method	Prefill (per token)		Decode (per step)	
	Arithmetic	Lookup	Arithmetic	Lookup
KIVI	$5d$	0	$5d + 2Ld$	0
QuaRot	$2d \log_2 h + 5d$	0	$2d \log_2 h + 5d + 2Ld$	0
OScaR (Ours)	$2d \log_2 h + 8d$	0	$2d \log_2 h + 8d + 3Ld$	0
TurboQuant (Original)	$6dh + 14.5d$	0	$6dh + 14.5d + Ld$	Ld
TurboQuant+	$4dh + 5.25d$	0	$4dh + 5.25d + Ld$	Ld

Table 6: Effective cost in millions of units: total prefill for L tokens and decode per step.

Method	Prefill total (M units)	Decode per step (M units)
KIVI	204.8	81.9
QuaRot	778.2	82.0
OScaR (Ours)	901.1	123.0
TurboQuant (Original)	32,051.0	249.0
TurboQuant+	21,187.0	247.9

L.1 Symbolic Operation Counts

We derive symbolic expressions for the computational cost of each method in terms of hidden dimension d, head dimension h, and sequence length L. The resulting symbolic operation counts are summarized in Table 5.

KIVI (Liu et al., 2024d). KIVI employs per-channel uniform quantization without query or key pre-transformation. During prefill, processing one token involves scanning all d dimensions to update per-channel min/max values, requiring $2d$ comparisons. Then, quantizing each element with $q = \text{round}((x - zero)/scale)$ incurs one subtraction, one division, and one rounding, totaling $3d$ operations. The prefill cost is therefore $5d$ arithmetic operations with no lookups. During decode, quantizing the newly generated key adds another $5d$ operations. Dequantizing the historical key cache of length L requires one subtraction and one multiplication per element, amounting to $2Ld$ operations. The total decode cost per step is $5d + 2Ld$ arithmetic operations.

QuaRot (Ashkboos et al., 2024). QuaRot applies an online Walsh-Hadamard Transform (WHT) to both query and key before quantization. For head dimension h, the WHT's butterfly structure requires $d \log_2 h$ additions per transformed tensor. During prefill, each token's query and key each undergo a WHT, contributing $2d \log_2 h$ additions, followed by the same $5d$ quantization as KIVI. The prefill cost is $2d \log_2 h + 5d$ arithmetic operations. During decode, the new key requires a WHT ($d \log_2 h$ adds) and quantization ($5d$ ops); the query requires a WHT ($d \log_2 h$ adds); and dequantizing the historical cache costs $2Ld$ ops. The total decode cost per step is $2d \log_2 h + 5d + 2Ld$ arithmetic operations.

OScaR (Ours). OScaR builds upon per-channel key quantization with two innovations: Canalized Rotation via online WHT and Omni-Token Scaling via token-wise L2 normalization. Canalized Rotation applies WHT to both query and key, requiring $d \log_2 h$ additions per tensor. Omni-Token Scaling normalizes each key token to unit length, involving three stages: (i) computing sum of squares across d dimensions ($2d$ operations), (ii) square root (1 operation), and (iii) scaling each element by the reciprocal of the norm (d multiplications). The total normalization cost is approximately $3d$ operations. During prefill, each token incurs Canalized Rotation on Q and K ($2d \log_2 h$ additions), Omni-Token Scaling on K ($3d$ ops), and per-channel quantization ($5d$ ops). Total prefill cost: $2d \log_2 h + 8d$ arithmetic operations, no lookups. During decode, the newly generated token requires: key rotation ($d \log_2 h$ adds), key scaling ($3d$ ops), key quantization ($5d$ ops), and query rotation ($d \log_2 h$ adds). Dequantizing the historical key cache costs $2Ld$ ops, and restoring key magnitudes via inverse scaling adds Ld multiplications. Total decode cost: $2d \log_2 h + 8d + 3Ld$ arithmetic operations, no lookups.

TurboQuant (Zandieh et al., 2025a) (Original). TurboQuant uses a 2.5-bit mixed-precision scheme (32 outlier channels at 3-bit, 96 normal channels at 2-bit) and a dense Haar QR rotation matrix $\Pi \in \mathbb{R}^{h \times h}$. Rotating a d-dimensional vector with this dense matrix requires $2dh$ arithmetic operations. During prefill, each token incurs: dense rotation of Q ($2dh$) and K ($2dh$), L2 normalization ($3d$), brute-force Lloyd-Max quantization ($7.5d$), residual handling (d), QJL projection ($2dh + d$), and residual norm ($2d$). Total: $6dh + 14.5d$ arithmetic operations. During decode, the new key requires dense rotation and quantization ($4dh + 14.5d$ ops); the query requires dense rotation ($2dh$ ops). Dequantizing the historical cache costs Ld arithmetic ops and Ld lookups. Total decode cost: $6dh + 14.5d + Ld$ arithmetic ops and Ld lookups.

TurboQuant+ (Turney & Contributors, 2026). This variant retains the 2.5-bit mixed-precision scheme, removes QJL, and replaces brute-force Lloyd-Max search with binary search, reducing quantization cost from $7.5d$ to $2.25d$ comparisons. No residual handling. During prefill: dense rotation of Q ($2dh$) and K ($2dh$), L2 normalization ($3d$), binary search quantization ($2.25d$). Total: $4dh + 5.25d$ arithmetic ops, no lookups. During decode: new key rotation and quantization ($2dh + 5.25d$ ops); query rotation ($2dh$ ops); dequantizing historical cache adds Ld arithmetic ops and Ld lookups. Total decode cost: $4dh + 5.25d + Ld$ arithmetic ops and Ld lookups.

L.2 Effective Cost Conversion

To enable fair comparison across methods with different operation types, we convert raw operation counts into weighted effective costs. We adopt a conservative weighting: one arithmetic operation costs 1 unit, and one random table lookup costs 5 units. This 1:5 ratio serves as a reasonable analytical compromise. On modern GPUs, arithmetic operations are heavily pipelined and can be issued at high throughput when data is in registers. In contrast, table lookups require address computation, memory accesses, and often suffer from irregular access patterns. Even under ideal L1 cache locality, a lookup typically incurs higher latency due to dependency stalls, and lookups can cause warp divergence. These factors collectively make table lookups more expensive than arithmetic operations.

We emphasize that this weighting is an approximation intended solely for comparative analysis. In real GPU execution, the true cost of random table lookups is often substantially higher than five arithmetic units. Nevertheless, this conservative weighting offers a transparent and reproducible basis for theoretical comparison. Accordingly, for a method with A arithmetic operations and T table lookups, its effective cost is $A + 5T$.

L.3 Numerical Results

For numerical illustration, we use an equal-width Query/Key cost model with hidden dimension $d = 4096$, head dimension $h = 128$ ($\log_2 h = 7$), and sequence length $L = 10{,}000$. This is an analytical configuration, not an exact accounting of the Qwen3-8B grouped-query kernels: unequal Query and KV widths require separate counts. Applying the 1:5 weighting to Table 5, and multiplying per-token prefill costs by L, gives Table 6.

L.4 Discussion

OScaR introduces token-wise normalization during prefill and decode, incurring a decode cost of 123.0 million units under the assumed configuration ($d = 4096$, $L = 10{,}000$). This overhead is offset by two key advantages: it eliminates all table lookups, avoiding irregular memory accesses, and relies solely on efficient operations, including fast Hadamard transforms and the hardware-accelerated `rsqrt` instruction. These theoretical estimates are validated by our empirical efficiency analysis in Section 4.3, where OScaR achieves substantial decode speedups while maintaining strong quantization fidelity with modest overhead.

M Implementation Details of OScaR's CUDA Kernels

This section outlines the key implementation details of the OScaR inference system. Our implementation builds upon two prior frameworks: BitDecoding (Du et al., 2025) for the 2-bit quantization and cache management backbone, and HadaCore (Agarwal et al., 2024) for Tensor-Core accelerated

Hadamard transforms. We extend both with fused Hadamard-norm preprocessing, integrated norm metadata, and residual-aware attention kernels.

M.1 Overall Design

OScaR compresses the KV cache to 2-bit representation to reduce memory footprint and bandwidth during long-context decoding. For Qwen3-8B, the core configuration is as follows: head dimension $d_h = 128$, group size $G = 32$ for per-channel key quantization, residual block size $R = 128$ for periodic cache flushing, and a K-side transformation consisting of Hadamard rotation followed by token-wise L2 normalization. Mathematically, the key is transformed as:

$$\mathbf{K}_h = \frac{\mathbf{H}(\mathbf{K})}{\sqrt{D}}, \qquad n_k = \|\mathbf{K}_h\|_2, \qquad \mathbf{K}_u = \frac{\mathbf{K}_h}{n_k},$$

where $D = 128$, $\mathbf{H}$ is the Hadamard matrix, and $\mathbf{K}_u$ is the unit direction vector stored in the low-bit cache. The token-wise norm n_k is stored as auxiliary metadata. During decoding, the query undergoes the same Hadamard rotation:

$$\mathbf{Q}_h = \frac{\mathbf{H}(\mathbf{Q})}{\sqrt{D}}, \qquad \text{logits} = \big(\mathbf{Q}_h \cdot \text{dequant}(\mathbf{K}_u)\big) \cdot n_k.$$

Because the Hadamard transform is orthogonal, the inner product is preserved, and the norm metadata restores the original magnitude of each key token.

M.2 Fused Hadamard-Norm Kernel

The fused preprocessing kernel operates on key tensors of shape $[\text{batch}, \text{seqlen}, n_{\text{kv_heads}}, d_h]$ with FP16 or BF16 data type. It outputs normalized keys and token-wise norms, combining the following steps into a single CUDA launch: reading all KV heads of a token, applying the normalized Hadamard transform to each head, accumulating squared sums across heads, computing the token-wise norm, and normalizing the transformed keys.

For generic implementations, one CUDA block processes one token with 128 threads, using butterfly Fast Hadamard Transform (FWHT) in shared memory followed by a reduction. For the fixed head dimension 128, we adopt the HadaCore-style Tensor Core optimization, which leverages the Kronecker decomposition $\mathbf{H}_{128} = \mathbf{H}_8 \otimes \mathbf{H}_{16}$. Specifically, we apply $\mathbf{H}_{16}$ via WMMA tiles and $\mathbf{H}_8$ via scalar butterfly, significantly reducing scalar instruction pressure compared to a naive FWHT.

The query side reuses the same kernel with Hadamard only (without normalization), ensuring that the inner product remains consistent with the original attention.

M.3 Quantization Format and Cache Organization

Key Quantization. Following BitDecoding, after Hadamard-norm preprocessing, $\mathbf{K}_u$ is quantized using per-channel grouping: each channel along the head dimension shares a scale and zero point across every $G = 32$ tokens. For 2-bit quantization, values range in $[0, 3]$, and eight values are packed into one uint16 word. A 128-token residual block contains exactly four quantization groups.

Value Quantization. Values are quantized via an offline Hadamard transform without requiring additional norm transformations, with per-token grouping along the head dimension, consistent with the original BitDecoding design. For $d_h = 128$ and $G = 32$, each token-head is partitioned into four quantization groups.

Cache Organization. The system maintains both packed and residual caches:

- **Packed cache:** $\mathbf{K}_u$ 2-bit payload, K scale and zero point, V 2-bit payload, V scale and zero point, and K token-wise norms.
- **Residual cache:** $\mathbf{K}_u$ residual (FP16), V residual (FP16), and K norm residual (FP16).

M.4 Prefill and Decode Workflows

Prefill. Attention computation itself remains unchanged and uses FlashAttention-2. After attention, the prompt sequence is split: the prefix (length a multiple of R) is quantized and stored in the packed cache after applying Hadamard-norm; the tail (length less than R) is stored in the residual cache without quantization.

Decode. Each new token undergoes projection, RMSNorm, RoPE, and Hadamard (for Q) or Hadamard-norm (for K). The transformed key, value, and norm are appended to the residual cache. The attention kernel `fwd_kvcache_int` processes both packed and residual caches as follows:

- For packed entries, keys are dequantized from their 2-bit representation and multiplied by their corresponding token-wise norms.
- For residual entries, keys are already in FP16 and are multiplied by their stored norms.

Both contributions are then combined in the logits before softmax and output projection. When the residual length reaches $R = 128$, the residual block is quantized and flushed to the packed cache.

Residual Flush. Flushing a 128-token block generates 16 packed rows $(128/8)$ and 4 parameter rows $(128/32)$. Since residual keys are already normalized, only quantization and packing are required, avoiding repeated Hadamard-norm computation.

M.5 Summary of Key Differences from the Baseline

Compared to the original BitDecoding, OScaR introduces the following key differences:

1. Hadamard rotation and token-wise normalization for keys, building upon HadaCore's efficient transform primitive.
2. Norm metadata incorporated into logits during attention.
3. Hadamard rotation for queries to preserve inner product semantics.
4. Removal of Hadamard and normalization on the value side.
5. A fixed 128-token residual flush, aligned with the 2-bit K-channel quantization groups.

M.6 Packed-cache storage and norm metadata

For the packed-cache formats evaluated in Table 1, INT2 payloads with one FP16 scale and one FP16 zero point per 32 values cost $2 + 32/32 = 3.0$ bits/value. The evaluated TurboQuant+ format costs $2.5 + 2 \times 32/128 = 3.0$ bits/value for mixed indices and two FP32 subset norms per 128-dimensional vector. These are nominal packed-cache rates; context-dependent residual windows and protected-token buffers are additional.

OScaR's system implementation stores one FP32 Key norm per token per layer, shared across KV heads. With $H_{\text{kv}} = 8$ and $d_h = 128$, this adds

$$\frac{32}{H_{\text{kv}} d_h} = 0.03125 \text{ bit/Key value}, \qquad \frac{32}{2H_{\text{kv}} d_h} = 0.015625 \text{ bit/KV value}. \tag{16}$$

Thus, including this norm gives 3.015625 bits/value for OScaR in this configuration, before residual buffers. Norm extraction, metadata writes, and inverse scaling are fused without a separate kernel launch.

Table 7: Nominal storage for the evaluated packed-cache formats, averaged over K/V. Residual windows and protected-token pools are excluded.

Methods	Included metadata	Bits/value
QuaRot / KIVI / OTT / RotateKV	FP16 scale and zero point, $G = 32$	3.0
TurboQuant+	Two FP32 subset norms, $d_h = 128$	3.0
OScaR	Scale/zero point plus shared FP32 Key norm	3.015625

N Details of Datasets and Benchmarks

N.1 Text-Only LLM Benchmarks

LongBench-E. LongBench (Bai et al., 2024) is a bilingual multitask benchmark for evaluating long-context understanding in large language models, covering both English and Chinese. It comprises 21 datasets across six task categories, with average lengths of 6,711 words for English and 13,386 characters for Chinese. The six categories are:

- **Single-Document QA (4 tasks):** MultiFieldQA-en, MultiFieldQA-zh, NarrativeQA, and Qasper. Models answer questions based on a single long document, testing detailed comprehension and information retrieval.
- **Multi-Document QA (4 tasks):** HotpotQA, 2WikiMultihopQA, MuSiQue, and DuReader. Models answer questions by synthesizing information across multiple documents, testing multi-hop reasoning.
- **Summarization (4 tasks):** GovReport, QMSum, MultiNews, and VCSUM. Models generate concise summaries of long documents or meeting transcripts, testing content selection and abstraction.
- **Few-Shot Learning (4 tasks):** TriviaQA, SAMSum, TREC, and LSHT. Models answer questions with few-shot examples provided in context, testing in-context learning ability.
- **Synthetic Tasks (3 tasks):** PassageRetrieval-en, PassageCount, and PassageRetrieval-zh. Models perform artificial tasks such as passage retrieval and counting, testing length generalization.
- **Code Completion (2 tasks):** LCC and RepoBench-P. Models predict the next line of code given long code contexts, including cross-file dependencies.

LongBench-E is a uniformly sampled subset where context lengths are evenly distributed across three intervals: 0–4k, 4k–8k, and 8k+ tokens. This design enables systematic analysis of model performance degradation as sequence length increases.

Needle-in-a-Haystack. The Needle-in-a-Haystack test (Kamradt, 2023) evaluates the in-context retrieval ability of long-context LLMs. A random fact (the needle) is inserted at varying depths within a long context window (the haystack), and the model is tasked with retrieving this specific statement. Performance is measured across context lengths ranging from 1K to 32K+ tokens and needle positions from 0% to 100% depth. The primary metric is retrieval accuracy, which reveals how well models maintain attention to relevant information buried in long sequences.

N.2 Multi-modal LLM Benchmarks

OCRBench. OCRBench (Liu et al., 2024c) is a comprehensive benchmark for evaluating optical character recognition capabilities in large multi-modal models. It comprises 29 datasets across five task categories, with a total of 1,000 human-annotated question-answer pairs:

- **Text Recognition (300 samples):** Recognizes regular, irregular, non-semantic, digit strings, handwriting, and artistic text from images.
- **Scene Text-Centric VQA (200 samples):** Answers questions about text appearing naturally in scenes, such as street signs, billboards, and storefronts, requiring both visual understanding and text reading.
- **Document-Oriented VQA (200 samples):** Focuses on structured documents such as forms, invoices, reports, and letters, requiring understanding of document layout, tables, and formatting.
- **Key Information Extraction (200 samples):** Extracts specific information, such as total amounts, dates, and names, from structured documents.
- **Handwritten Mathematical Expression Recognition (100 samples):** Recognizes and transcribes handwritten mathematical formulas into LaTeX format.

Table 8: OCRBench evaluation results comparing TurboQuant+ with and without QJL. TurboQuant+ uses a 2.5-bit setting, whereas OScaR employs INT2 quantization with a group size of 128.

Model	Method	Recog.	VQA^S	VQA^D	KIE	HMER	Final Score
Sample Size		300	200	200	200	100	1000
LLaVA-v1.6-vicuna-7B	16bit	187	160	82	107	0	536
	TurboQuant+ (w/ QJL)	161	144	41	57	0	403
	TurboQuant+ (w/o QJL)	173	156	73	99	0	501
	OScaR (ours)	178	158	82	101	0	519
Qwen3-VL-8B	16bit	270	181	174	181	52	858
	TurboQuant+ (w/ QJL)	266	136	57	143	24	626
	TurboQuant+ (w/o QJL)	269	178	169	182	49	847
	OScaR (ours)	270	180	170	183	53	856

DocVQA. DocVQA (Mathew et al., 2021) is a dataset for visual question answering on document images. It contains over 12,000 document images with 50,000 questions, covering diverse document types including forms, invoices, reports, letters, and tables. Questions require understanding of document structure, layout, and visual elements beyond simple text extraction. The validation set comprises 5,349 samples. The primary evaluation metric is ANLS (Average Normalized Levenshtein Similarity), which accounts for minor OCR and spelling variations.

N.3 Omni-modal LLM Benchmark

MMAU-Pro. MMAU-Pro (Kumar et al., 2026) is a comprehensive benchmark for holistic evaluation of audio intelligence in AI systems. It contains 5,305 instances, each comprising one or more audio clips paired with human expert-generated question-answer pairs, spanning speech, non-speech sounds, music, and their combinations. The benchmark evaluates auditory intelligence across 49 unique skills and multiple complex dimensions, including long-form audio comprehension, spatial audio reasoning, and multi-audio understanding. All questions are designed to require deliberate multi-hop reasoning and include both multiple-choice and open-ended response formats. Notably, audio data is sourced directly from real-world environments ("from the wild") rather than from existing datasets with known distributions.

For our evaluation, we focus on two challenging subsets that demand reasoning beyond standard multiple-choice: open-ended QA and audio instruction following. The open-ended subset requires models to generate free-form responses. Following the MMAU-Pro protocol, we evaluate these responses using Qwen2.5-7B-Instruct as an LLM judge, which scores each response from 1 to 5 across four criteria: correctness, relevance, completeness, and clarity. The scores are then converted to percentages for consistent comparison with multiple-choice results. The instruction-following subset comprises constraint instances drawn from 28 instruction types, with responses evaluated using deterministic scripts.

O Additional TurboQuant+ Implementation Details

Since TurboQuant does not provide an official code release, we adopt TurboQuant+ (Turney & Contributors, 2026), a widely used open-source implementation, for all evaluations. TurboQuant+ officially omits the QJL step (Zandieh et al., 2025b), as it has been shown to increase variance that is subsequently amplified by the softmax function, ultimately degrading generation quality. To ensure a more challenging comparison, we also exclude the QJL step in our experiments, as it empirically reduces model performance (see Table 8).

We also account for the average bit overhead introduced by quantization parameters in OScaR and other comparison methods. Specifically, TurboQuant+ employs a mixed-precision 2.5-bit quantization scheme: outlier channels are assigned higher bit-widths to preserve fidelity, while regular channels use 2 bits, yielding an average of approximately 2.5 bits per element. All other methods adopt INT2 quantization. Furthermore, since most competing baselines apply the same quantization configuration across layers, we adopt a consistent setup for every layer in TurboQuant+, uniformly quantizing both Key and Value without layer-wise adaptive precision.

Table 9: OCRBench evaluation results across six task categories: Recognition (Recog.), Scene Text VQA (VQA^S), Document Text VQA (VQA^D), Key Information Extraction (KIE), Handwritten Mathematical Expression Recognition (HMER), and the final weighted score. The superscripts ^{S} and ^{D} denote scene text and document text, respectively. All competing methods except TurboQuant+ are configured with INT2 quantization and a group size of 128. TurboQuant+ uses a 2.5-bit setting. TurboQuant is based on TurboQuant+ (Turney & Contributors, 2026); QJL is excluded as it degrades performance. See Appendix O for details.

Model	**Method**	**Recog.**	***VQA*^{S}**	***VQA*^{D}**	**KIE**	**HMER**	**Final Score**
Sample Size		300	200	200	200	100	1000
LLaVA-v1.6-vicuna-7B	16bit	187	160	82	107	0	536
	QuaRot	177	152	61	91	0	481
	RotateKV	168	152	64	89	0	473
	KIVI	173	157	69	89	0	488
	OTT	**187**	157	72	97	0	513
	TurboQuant+	173	156	73	99	0	501
	OScaR (ours)	178	**158**	**82**	**101**	0	**519**
Qwen3-VL-8B	16bit	270	181	174	181	52	858
	QuaRot	266	171	104	150	31	722
	RotateKV	267	163	127	162	35	754
	KIVI	268	178	**173**	180	52	851
	OTT	**270**	179	170	177	54	850
	TurboQuant+	269	178	169	182	49	847
	OScaR (ours)	**270**	**180**	170	**183**	**53**	**856**
Qwen3-VL-4B	16bit	261	183	171	186	51	852
	QuaRot	254	173	149	159	38	773
	RotateKV	254	143	84	134	23	638
	KIVI	255	177	160	175	46	813
	OTT	256	180	167	177	**51**	831
	TurboQuant+	**260**	**183**	**170**	175	40	828
	OScaR (ours)	259	181	169	**181**	48	**838**

P Experimental Results and Analysis on Needle-in-a-Haystack

In this section, we present the Needle-in-a-Haystack experiment (Kamradt, 2023). Following the standard protocol, we evaluate retrieval accuracy across context lengths up to 42,000 tokens and at 15 different needle depth positions. A detailed description of the benchmarks is provided in Appendix N. The results are shown in Figure 29. The 16-bit full-precision baseline is provided for reference. OScaR achieves the highest retrieval accuracy among all quantized methods (96.5%), slightly surpassing the 16-bit baseline (96.0%) and outperforming the second-best quantized method (92.7%).

Q Experimental Results and Analysis on OCRBench

Table 9 reports results on the OCRBench benchmark (Liu et al., 2024c). The final score reflects the number of correctly answered samples. A detailed dataset description is provided in Appendix N. OScaR consistently achieves the highest accuracy among all 2-bit quantization methods across all three evaluated models. On LLaVA-v1.6-vicuna-7B, OScaR attains 51.9% accuracy, outperforming the second-best method by 0.6 percentage points. On Qwen3-VL-8B, OScaR achieves 85.6% accuracy, coming within 0.2 percentage points of the 16-bit baseline (85.8%). On Qwen3-VL-4B, OScaR scores 83.8% accuracy, surpassing the best competing 2-bit method by 0.7 percentage points. Overall, OScaR delivers robust performance on multi-modal document understanding tasks under INT2 KV cache quantization.

Table 10: DocVQA evaluation results. All competing methods except TurboQuant+ are configured with INT2 quantization and a group size of 128. TurboQuant+ uses a 2.5-bit setting. TurboQuant is based on TurboQuant+ (Turney & Contributors, 2026); QJL is excluded as it degrades performance. See Appendix O for details.

Method	16bit	QuaRot	RotateKV	KIVI	OTT	TurboQuant+	OScaR (ours)
LLaVA-v1.6-vicuna-7B	68.57	65.55	65.26	65.34	66.11	67.09	**67.79**
Qwen3-VL-8B	94.93	93.93	92.76	94.70	94.86	94.51	**95.01**
Qwen3-VL-4B	94.23	89.89	76.12	93.49	93.24	93.54	**93.85**

R Experimental Results and Analysis on DocVQA

Table 10 reports results on the DocVQA benchmark (Mathew et al., 2021), which evaluates document understanding through visual question answering. A detailed description of the dataset is provided in Appendix N.

OScaR consistently achieves the highest accuracy among all 2-bit quantization methods across the three evaluated models. On Qwen3-VL-8B, it slightly surpasses the 16-bit baseline, demonstrating near-lossless performance under INT2 quantization. On Qwen3-VL-4B, OScaR trails the 16-bit baseline by only 0.4 percentage points while outperforming the strongest competing 2-bit method by 0.36 percentage points. On LLaVA-v1.6-vicuna-7B, it exceeds the best quantized baseline by 0.7 percentage points and remains within 1.1 percentage points of the 16-bit reference. These results confirm that OScaR's *Canalized Rotation* and *Omni-Token Scaling* generalize effectively across diverse multi-modal models and tasks.

S Experimental Results and Analysis on MMAU-Pro

For our evaluation on MMAU-Pro, we focus on two challenging subsets that demand reasoning beyond standard multiple-choice: open-ended QA and audio instruction following.

The open-ended subset requires models to generate free-form responses. Following the MMAU-Pro protocol, we evaluate these responses using Qwen2.5-7B-Instruct as an LLM judge, which scores each response from 1 to 5 across four criteria: correctness, relevance, completeness, and clarity. The scores are then converted to percentages for consistent comparison with multiple-choice results. We additionally report the "Good Rate," defined as the proportion of samples where the LLM judge assigns an overall score of 4.0 or higher, indicating high-quality responses. The instruction-following subset comprises constraint instances drawn from 28 instruction types, with responses evaluated using deterministic scripts.

Table 11: MMAU-Pro evaluation results. "Good Rate" denotes the proportion of open-ended responses with an overall score of 4 or higher (out of 5), and "AIF" stands for Audio Instruction Following accuracy. All competing methods except TurboQuant+ are configured with INT2 quantization and a group size of 128. TurboQuant+ uses a 2.5-bit setting. TurboQuant is based on TurboQuant+ (Turney & Contributors, 2026); QJL is excluded as it degrades performance. See Appendix O for details.

Qwen3-Omni-30B-A3B	16bit	KIVI	OTT	TurboQuant+	OScaR (ours)
Open-ended	66.2	65.8	65.8	66.6	**67.4**
Good Rate	27.8	27.0	26.9	27.0	**29.8**
AIF	87.4	78.2	83.9	79.3	**88.5**

As shown in Table 11, OScaR achieves the highest scores across all three evaluation metrics. On the open-ended subset, OScaR slightly surpasses the 16-bit baseline by 1.2 percentage points and outperforms all other quantized methods. More notably, OScaR achieves a Good Rate that exceeds the 16-bit baseline by 2.0 percentage points and outperforms the best competing quantized method by 2.8 percentage points, indicating that OScaR preserves high-quality response generation under

extreme compression. For audio instruction following (AIF), OScaR surpasses the 16-bit baseline by 1.1 percentage points, demonstrating near-lossless performance. These results demonstrate OScaR's strong generalization capability across omni-modal tasks and models, maintaining superior performance under INT2 quantization on the challenging MMAU-Pro benchmark.

T TNI Analysis Before and After OScaR

To empirically verify the effectiveness of OScaR in mitigating TNI, we visualize token norm distributions across several models before and after applying OScaR. As shown in Figures 25, 26, 27 and 28, OScaR consistently alleviates TNI, transforming scattered norm distributions into more compact and balanced patterns across different models and modalities. These visualizations provide strong empirical support for OScaR's superior performance under extreme KV cache quantization.

U Ablation Study

In this section, we present ablation studies examining the contributions of each proposed component. Table 12 dissects the effect of our two core innovations: *Canalized Rotation* and *Omni-Token Scaling*. Applying *Omni-Token Scaling* after *Canalized Rotation* substantially recovers accuracy from the severely degraded INT2 baseline. In contrast, applying direct token-wise scaling alone without *Canalized Rotation* further harms performance, aligning with the analysis in Section 3.2.

Table 12: Ablation of core components in OScaR on the WorldSense benchmark (Hong et al., 2025). "INT2 KCVT" denotes per-channel Key and per-token Value quantization with 2-bit integers. The quantization group size is configured as 128. The complete OScaR configuration (last row) achieves the best trade-off, recovering accuracy from the degraded INT2 baseline.

WorldSense			**Qwen2.5-Omni-3B**	**Qwen2.5-Omni-7B**
+ INT2 KCVT			22.64	39.62
+ INT2 KCVT		+ Omni-Token Scaling	17.61	35.22
+ INT2 KCVT	+ Canalized Rotation		37.74	41.51
+ INT2 KCVT	**+ Canalized Rotation**	**+ Omni-Token Scaling**	**38.36**	**42.77**

Conditional complementarity. On Qwen2.5-Omni-7B, OTS changes plain INT2 accuracy from 39.62 to 35.22, whereas adding OTS after CR changes 41.51 to 42.77. This reversal of OTS's marginal effect supports conditional complementarity: rotation preserves token norms, while scaling requires suitable channel geometry to avoid the artifact analyzed in Appendix J.

We further investigate alternative normalization strategies for computing the scaling coefficient in *Omni-Token Scaling*:

- ℓ_2 **norm:** uses the Euclidean norm of each token's hidden representation.
- **Rsqrt:** employs the hardware-accelerated reciprocal square root instruction to efficiently approximate $1/\sqrt{\sum x_i^2}$, offering faster computation than explicit ℓ_2 norm calculation.
- **Max:** uses the maximum absolute value across channels of each token.
- **Mean absolute value:** averages the absolute values across channels.

Table 13: Comparison of alternative scaling coefficient strategies for *Omni-Token Scaling* on the LongBench-E benchmark (Bai et al., 2024).

LongBench-E	**ℓ_2 norm**	**Rsqrt**	**Max**	**Mean absolute value**
Llama-2-7B	29.59	29.57	29.23	29.68
Llama-3-8B	38.56	38.63	33.68	38.53
Qwen2.5-7B	42.57	42.83	14.47	41.95
Qwen3-8B	42.85	42.91	25.12	42.64

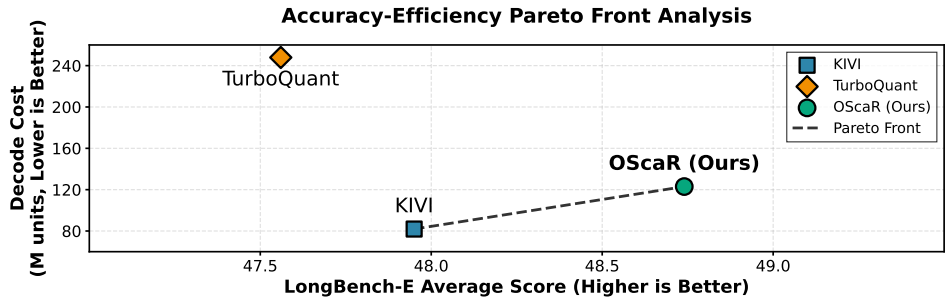


Figure 9: Pareto front analysis of KV cache quantization methods on Qwen3-8B. The x-axis represents the average LongBench-E accuracy (higher is better), and the y-axis represents the decode cost in million units (lower is better). OScaR achieves the highest accuracy with competitive efficiency, occupying a distinct and advantageous position on the Pareto front.

As shown in Table 13, the ℓ_2 norm and its rsqrt approximation achieve the best and mutually comparable performance across all evaluated model families. Heuristics such as Max cause severe degradation: on Qwen2.5-7B, Max collapses accuracy to 14.47 compared to 42.57 with the ℓ_2 norm. The mean absolute value, while not optimal, performs competitively and avoids such catastrophic drops. Given the negligible difference between ℓ_2 norm and rsqrt, we adopt the rsqrt-based implementation in our final setup due to its superior hardware efficiency and lower latency.

V Accuracy-Efficiency Pareto Front Analysis

In this section, we analyze the trade-off between computational efficiency and task accuracy using the theoretical decode cost per step from Table 6 (Appendix L) and the LongBench-E scores of Qwen3-8B from Table 1. We compare OScaR against several state-of-the-art methods, including KIVI and TurboQuant+ (as detailed in Appendix O). Figure 9 visualizes the accuracy against decode cost for each method.

- **KIVI** establishes the efficiency baseline. With the lowest decode cost among all methods, it incurs no rotation or normalization overhead, achieving an accuracy of 47.95. This reflects the inherent trade-off of minimal preprocessing.
- **TurboQuant+** has a decode cost of 247.9 million units, compared with 81.9 for KIVI and 123.0 for OScaR (Table 6). Its LongBench-E average is 47.56, below KIVI's 47.95, so KIVI dominates it in this analytical cost–accuracy comparison.
- **OScaR** achieves a favorable balance between efficiency and accuracy. Its decode cost is 1.5 times that of KIVI but less than half that of TurboQuant+, while delivering the highest accuracy among all quantized methods.

Overall, OScaR occupies a distinct and advantageous position on the Pareto front, offering a favorable combination of competitive computational cost and strong accuracy.

W Additional Decoding Efficiency Measurements

Table 14 reports the decoding latency of OScaR and TurboQuant+ alongside the `BF16 FlashDecoding-v2` baseline across context lengths ranging from 1K to 128K tokens. OScaR is implemented on the PyTorch runtime, while TurboQuant+ relies on `llama.cpp`; this comparison measures system performance and does not isolate algorithmic overhead. OScaR exhibits remarkably stable performance across the entire range, with latency increasing only modestly from 24.1 ms/tok at 16K to 30.9 ms/tok at 128K. In contrast, TurboQuant+ shows strong context dependence: while it achieves lower latency than OScaR at short contexts (e.g., 7.8 ms/tok at 1K), its latency grows rapidly, reaching 72.9 ms/tok at 128K.

Table 14: Decode latency comparison on Qwen3-8B with a single H20 GPU (141GB), measured in milliseconds per token.

Context Length	FlashDecoding-v2 (ms/tok)	OScaR (ms/tok)	TurboQuant+ (ms/tok)
1K	19.5	25.1	7.8
2K	19.8	26.4	8.5
4K	20.3	24.9	9.6
8K	23.8	24.9	11.7
16K	28.3	24.1	15.7
32K	38.0	25.8	23.9
48K	47.1	25.7	32.1
64K	56.3	25.3	40.2
96K	74.6	28.5	56.4
128K	92.9	30.9	72.9

OScaR is slower than BF16 at short contexts but becomes faster as KV-cache traffic grows. At 128K tokens, OScaR achieves a 3.0× speedup (92.9 ms/tok vs. 30.9 ms/tok), whereas TurboQuant+ attains only a modest 1.3× speedup at the same context length.

W.1 Controlled end-to-end and component-wise latency

We benchmark Qwen3-8B on one A800 80GB GPU at batch size 1 with identical inputs and positions, generating 320 tokens at 4K, 16K, 64K, and 128K contexts. Values are medians of five requests after warmup. Time to first token (TTFT) includes prefill, KV packing, the LM head, and first-token selection; time per output token (TPOT) averages the subsequent 319 steps, including cache update/flush and attention.

The KIVI-style INT2 control uses per-channel Keys and per-token Values on the same BitDecoding backend as OScaR, with CR and OTS disabled; it is not the official KIVI runtime. CR-only and OTS-only enable each component separately. OTS-only attributes cost; its accuracy limitation appears in Table 12.

Table 15: Controlled end-to-end latency on an A800 80GB GPU. Each cell is TTFT / TPOT in milliseconds.

Context	BF16 FlashDecoding	KIVI-style INT2	OScaR INT2
4K	366.30 / 28.94	396.38 / 32.25	400.25 / 34.09
16K	1,695.18 / 29.80	1,772.77 / 32.15	1,790.27 / 33.89
64K	11,649.19 / 54.29	11,881.14 / 34.36	11,882.33 / 35.38
128K	37,786.62 / 91.40	38,125.83 / 47.89	37,921.31 / 48.90

Table 16: Component-wise latency on the same A800/BitDecoding setup. Each cell is TTFT / TPOT in milliseconds.

Context	KIVI-style INT2	CR-only	OTS-only	OScaR
4K	396.38 / 32.25	403.20 / 33.07	403.51 / 33.86	400.25 / 34.09
16K	1,772.77 / 32.15	1,817.04 / 33.13	1,820.35 / 33.61	1,790.27 / 33.89
64K	11,881.14 / 34.36	12,192.38 / 35.08	12,221.89 / 35.01	11,882.33 / 35.38
128K	38,125.83 / 47.89	39,580.81 / 48.56	39,073.33 / 48.40	37,921.31 / 48.90

We use TPOT to attribute online overhead because TTFT includes full prefill and varies more between runs. Relative to the KIVI-style control, CR-only adds 2.5%/3.1%/2.1%/1.4% TPOT at 4K/16K/64K/128K; OTS-only adds 5.0%/4.5%/1.9%/1.1%; and full OScaR adds 5.7%/5.4%/3.0%/2.1%. Shared fused paths mean isolated costs need not add linearly. OTS-only jointly measures normalization, metadata handling, and inverse scaling, rather than timing each operation separately.

At 4K/16K, the fixed INT2 kernel costs are not yet amortized. At 64K/128K, OScaR reduces TPOT relative to BF16 by 34.8%/46.5%; at 128K its TTFT is within 0.4% of BF16. These A800 measurements are separate from the H20 measurements in Section 4.3 and Table 14.

Algorithm 1: The OScaR algorithm.

parameter: group size G, residual length R, head dimension d_h

Procedure `OScaR-Preprocess`**:**

Input: $\mathbf{W}_V \in \mathbb{R}^{d_h \times d_h}$, $\mathbf{W}_O \in \mathbb{R}^{d_h \times d_h}$

$\mathbf{H} \leftarrow$ Hadamard matrix of size $d_h \times d_h$

$\mathbf{W}_V \leftarrow \mathbf{W}_V\mathbf{H}, \quad \mathbf{W}_O \leftarrow \mathbf{H}\mathbf{W}_O$

return $\mathbf{W}_V, \mathbf{W}_O$

end

Procedure `OScaR-Inference`**:**

Input: Input $\mathbf{X}$, KV cache (empty for prefill)

Output: Output $\mathbf{o}$

$\mathbf{X}_Q = \mathbf{X}\mathbf{W}_Q, \quad \mathbf{X}_K = \mathbf{X}\mathbf{W}_K, \quad \mathbf{X}_V = \mathbf{X}\mathbf{W}_V$

$\mathbf{X}_Q \leftarrow \text{FHT}(\mathbf{X}_Q), \quad \mathbf{X}_K \leftarrow \text{FHT}(\mathbf{X}_K)$

$\mathbf{s}_K \leftarrow \|\mathbf{X}_K\|_2, \quad \mathbf{X}_K \leftarrow \mathbf{X}_K/\mathbf{s}_K$

$Q(\mathbf{X}_K^{hist}), \mathbf{X}_{K_r}^{hist}, \mathbf{s}_{K_g}^{hist}, \mathbf{s}_{K_r}^{hist}, Q(\mathbf{X}_V^{hist}), \mathbf{X}_{V_r}^{hist} \leftarrow$ `KV cache`

$\mathbf{K}_{all} \leftarrow \text{Concat}([\text{Dequant}(Q(\mathbf{X}_K^{hist})) \cdot \mathbf{s}_{K_g}^{hist},\ \mathbf{X}_{K_r}^{hist} \cdot \mathbf{s}_{K_r}^{hist},\ \mathbf{X}_K \cdot \mathbf{s}_K])$

$\mathbf{V}_{all} \leftarrow \text{Concat}([\text{Dequant}(Q(\mathbf{X}_V^{hist})),\ \mathbf{X}_{V_r}^{hist},\ \mathbf{X}_V])$

$\mathbf{o} \leftarrow \text{Attention}(\mathbf{X}_Q, \mathbf{K}_{all}, \mathbf{V}_{all})\mathbf{W}_O$

$Q(\mathbf{X}_K^{curr}), \mathbf{X}_{K_r}^{curr}, \mathbf{s}_{K_g}^{curr}, \mathbf{s}_{K_r}^{curr} \leftarrow$ `BufferQuantK`$(\mathbf{X}_K, \mathbf{s}_K, \mathbf{X}_{K_r}^{hist}, \mathbf{s}_{K_r}^{hist})$

$Q(\mathbf{X}_V^{curr}), \mathbf{X}_{V_r}^{curr} \leftarrow$ `BufferQuantV`$(\mathbf{X}_V, \mathbf{X}_{V_r}^{hist})$

`KV cache` $\leftarrow Q(\mathbf{X}_K^{curr}), \mathbf{X}_{K_r}^{curr}, \mathbf{s}_{K_g}^{curr}, \mathbf{s}_{K_r}^{curr}, Q(\mathbf{X}_V^{curr}), \mathbf{X}_{V_r}^{curr}$

return $\mathbf{o}$

end

Function `BufferQuantK`$(\mathbf{M}, \mathbf{s}, \mathbf{M}_r, \mathbf{s}_r)$**:**

if *prefill (cache empty)* **then**

$r = \text{len}(\mathbf{M})\%R$

$\mathbf{M}_g \leftarrow \mathbf{M}[:-r], \quad \mathbf{M}_r \leftarrow \mathbf{M}[-r:]$

$\mathbf{s}_g \leftarrow \mathbf{s}[:-r], \quad \mathbf{s}_r \leftarrow \mathbf{s}[-r:]$

$\mathbf{M}_g^{quant} \leftarrow \text{GroupQuant}(\mathbf{M}_g, \text{dim=channel}, \text{numGroup} = \text{len}(\mathbf{M}_g)//G)$

return $\mathbf{M}_g^{quant}, \mathbf{M}_r, \mathbf{s}_g, \mathbf{s}_r$

else

Append $\mathbf{M}$ to $\mathbf{M}_r$, append $\mathbf{s}$ to $\mathbf{s}_r$

if *len*$(\mathbf{M}_r) = R$ **then**

$\mathbf{M}_r^{quant} \leftarrow \text{GroupQuant}(\mathbf{M}_r, \text{dim=channel}, \text{numGroup} = R//G)$

$\mathbf{M}_g \leftarrow \text{Concat}(\mathbf{M}_g, \mathbf{M}_r^{quant}), \quad \mathbf{s}_g \leftarrow \text{Concat}(\mathbf{s}_g, \mathbf{s}_r)$

$\mathbf{M}_r \leftarrow \emptyset, \quad \mathbf{s}_r \leftarrow \emptyset$

end

return $\mathbf{M}_g, \mathbf{M}_r, \mathbf{s}_g, \mathbf{s}_r$

end

Function `BufferQuantV`$(\mathbf{M}, \mathbf{M}_r)$**:**

if *prefill (cache empty)* **then**

$r = \text{len}(\mathbf{M})\%R$

$\mathbf{M}_g \leftarrow \mathbf{M}[:-r], \quad \mathbf{M}_r \leftarrow \mathbf{M}[-r:]$

$\mathbf{M}_g^{quant} \leftarrow \text{GroupQuant}(\mathbf{M}_g, \text{dim=token}, \text{numGroup} = d_h//G)$

return $\mathbf{M}_g^{quant}, \mathbf{M}_r$

else

Append $\mathbf{M}$ to $\mathbf{M}_r$

if *len*$(\mathbf{M}_r) = R$ **then**

$\mathbf{M}_r^{quant} \leftarrow \text{GroupQuant}(\mathbf{M}_r, \text{dim=token}, \text{numGroup} = d_h//G)$

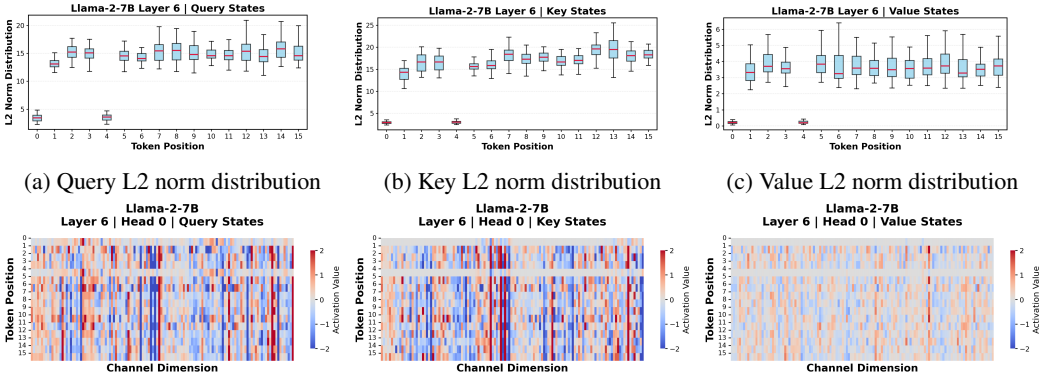
$\mathbf{M}_g \leftarrow \text{Concat}(\mathbf{M}_g, \mathbf{M}_r^{quant})$

$\mathbf{M}_r \leftarrow \emptyset$

end

return $\mathbf{M}_g, \mathbf{M}_r$

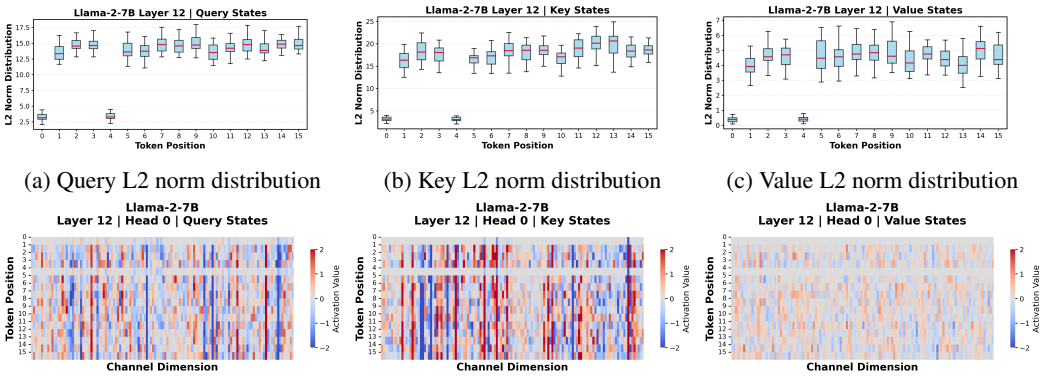
end



(a) Query L2 norm distribution (b) Key L2 norm distribution (c) Value L2 norm distribution

(d) Query heatmap (e) Key heatmap (f) Value heatmap

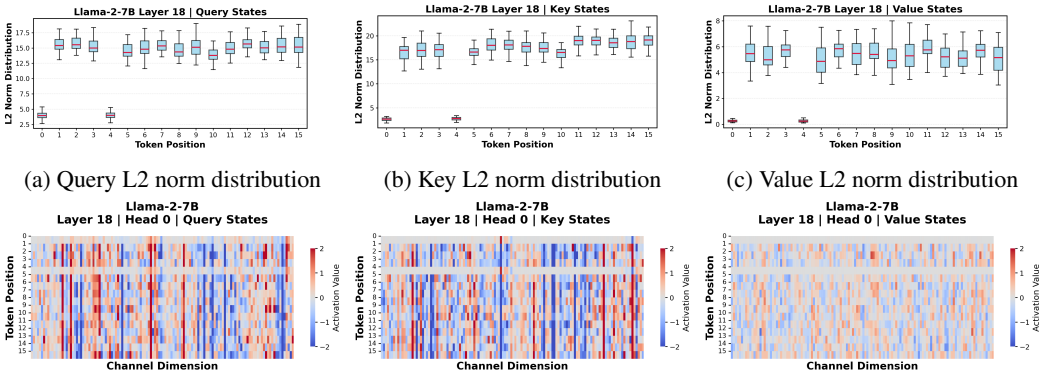
Figure 10: L2 norm distributions (top row) and value heatmaps (bottom row) of Query, Key, and Value states in Layer 6 of Llama-2-7B.



(a) Query L2 norm distribution (b) Key L2 norm distribution (c) Value L2 norm distribution

(d) Query heatmap (e) Key heatmap (f) Value heatmap

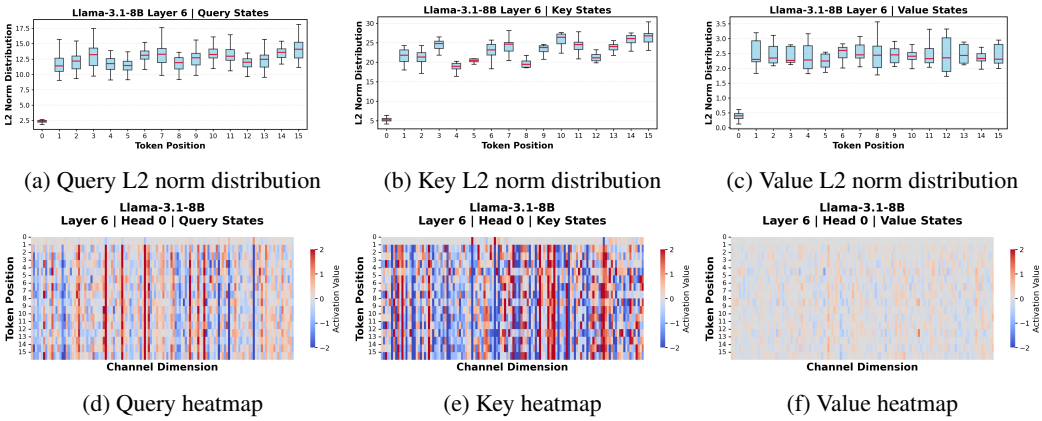
Figure 11: L2 norm distributions (top row) and value heatmaps (bottom row) of Query, Key, and Value states in Layer 12 of Llama-2-7B.



(a) Query L2 norm distribution (b) Key L2 norm distribution (c) Value L2 norm distribution

(d) Query heatmap (e) Key heatmap (f) Value heatmap

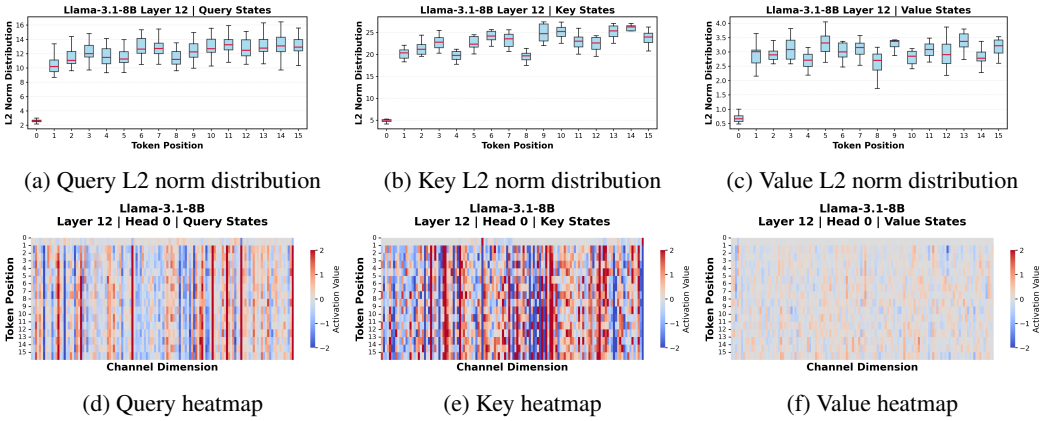
Figure 12: L2 norm distributions (top row) and value heatmaps (bottom row) of Query, Key, and Value states in Layer 18 of Llama-2-7B.



(a) Query L2 norm distribution (b) Key L2 norm distribution (c) Value L2 norm distribution

(d) Query heatmap (e) Key heatmap (f) Value heatmap

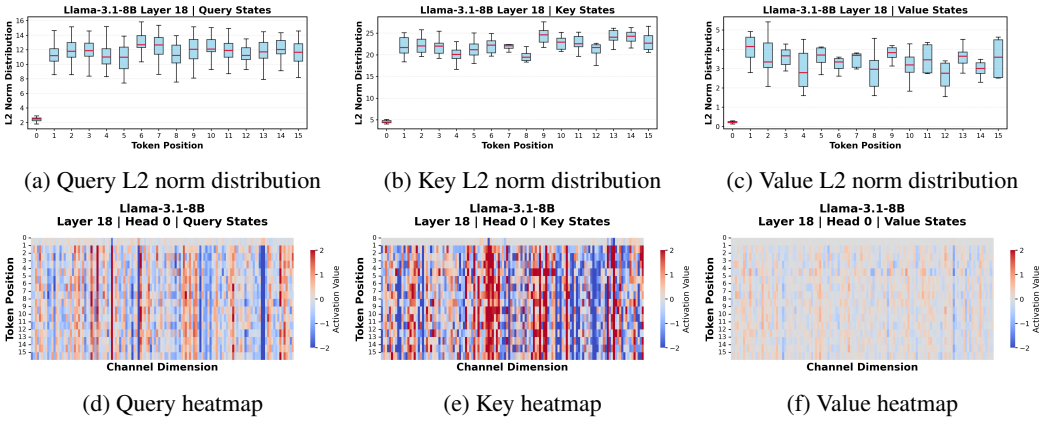
Figure 13: L2 norm distributions (top row) and value heatmaps (bottom row) of Query, Key, and Value states in Layer 6 of Llama-3.1-8B.



(a) Query L2 norm distribution (b) Key L2 norm distribution (c) Value L2 norm distribution

(d) Query heatmap (e) Key heatmap (f) Value heatmap

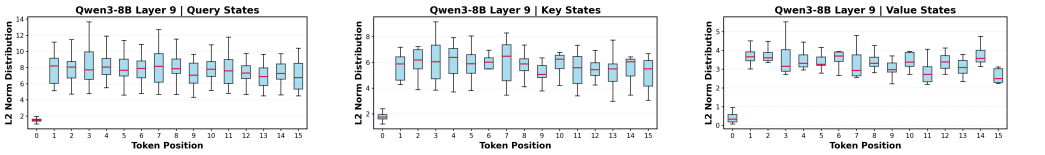
Figure 14: L2 norm distributions (top row) and value heatmaps (bottom row) of Query, Key, and Value states in Layer 12 of Llama-3.1-8B.



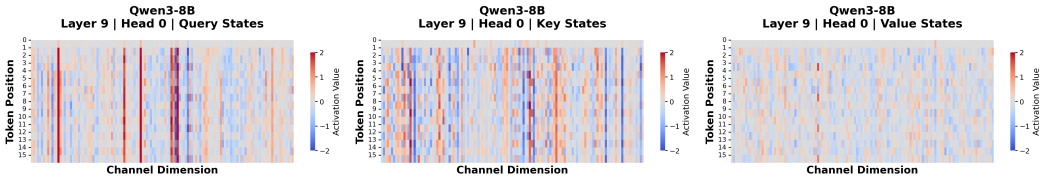
(a) Query L2 norm distribution (b) Key L2 norm distribution (c) Value L2 norm distribution

(d) Query heatmap (e) Key heatmap (f) Value heatmap

Figure 15: L2 norm distributions (top row) and value heatmaps (bottom row) of Query, Key, and Value states in Layer 18 of Llama-3.1-8B.

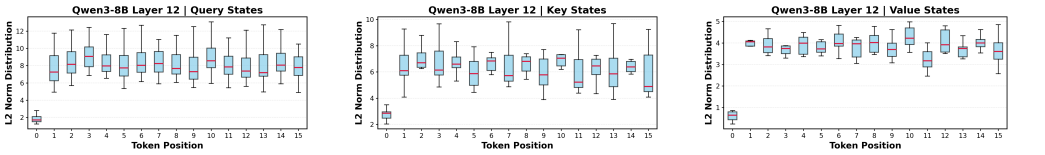


(a) Query L2 norm distribution (b) Key L2 norm distribution (c) Value L2 norm distribution

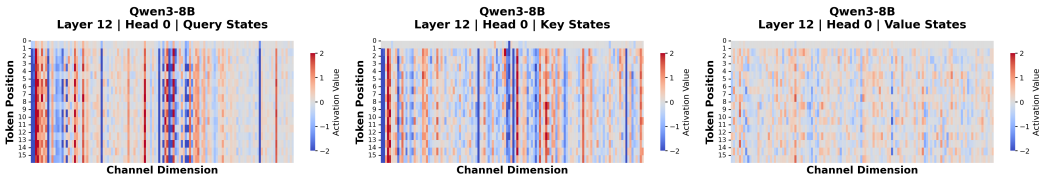


(d) Query heatmap (e) Key heatmap (f) Value heatmap

Figure 16: L2 norm distributions (top row) and value heatmaps (bottom row) of Query, Key, and Value states in Layer 9 of Qwen-3-8B.

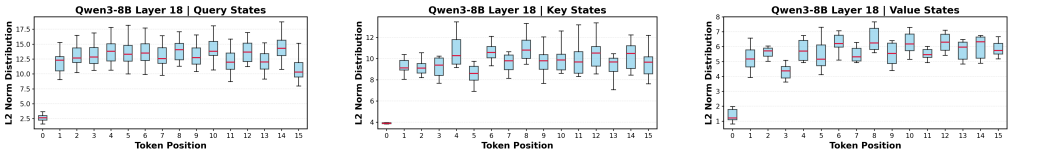


(a) Query L2 norm distribution (b) Key L2 norm distribution (c) Value L2 norm distribution

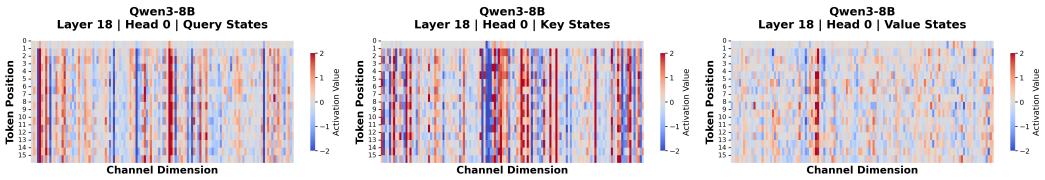


(d) Query heatmap (e) Key heatmap (f) Value heatmap

Figure 17: L2 norm distributions (top row) and value heatmaps (bottom row) of Query, Key, and Value states in Layer 12 of Qwen-3-8B.

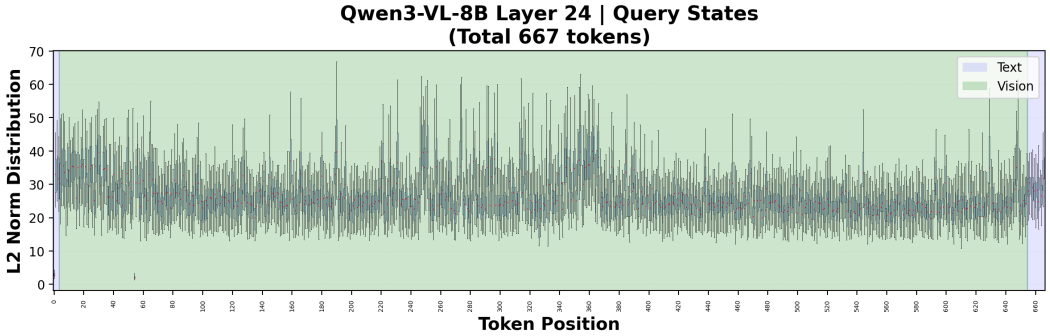


(a) Query L2 norm distribution (b) Key L2 norm distribution (c) Value L2 norm distribution

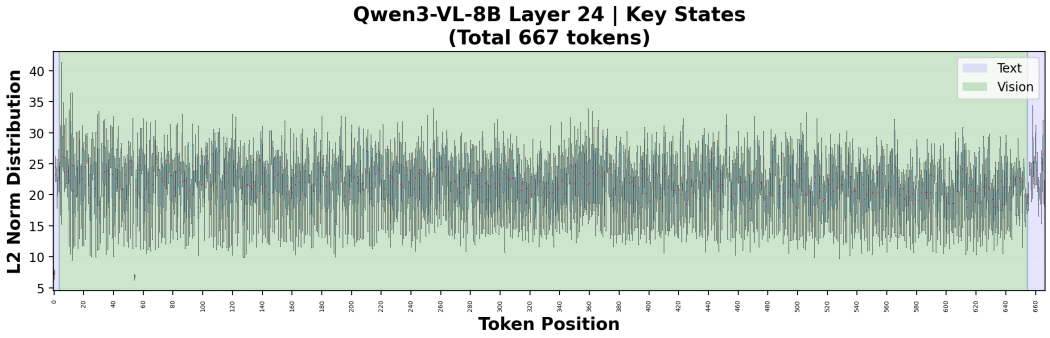


(d) Query heatmap (e) Key heatmap (f) Value heatmap

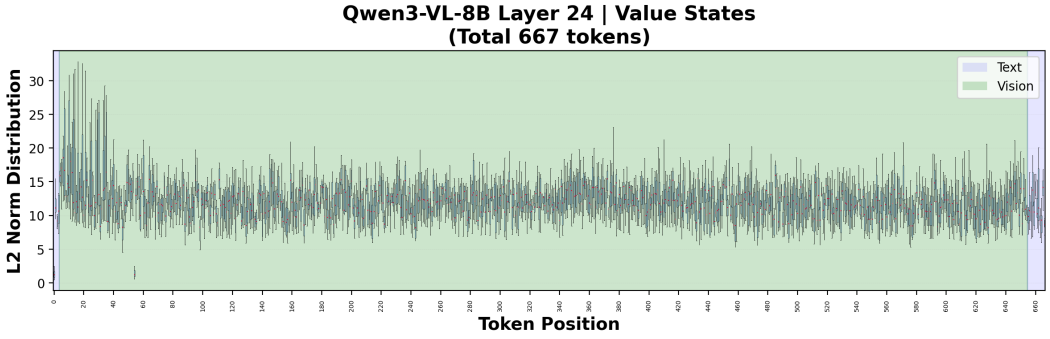
Figure 18: L2 norm distributions (top row) and value heatmaps (bottom row) of Query, Key, and Value states in Layer 18 of Qwen-3-8B.



(a) Query L2 norm distribution

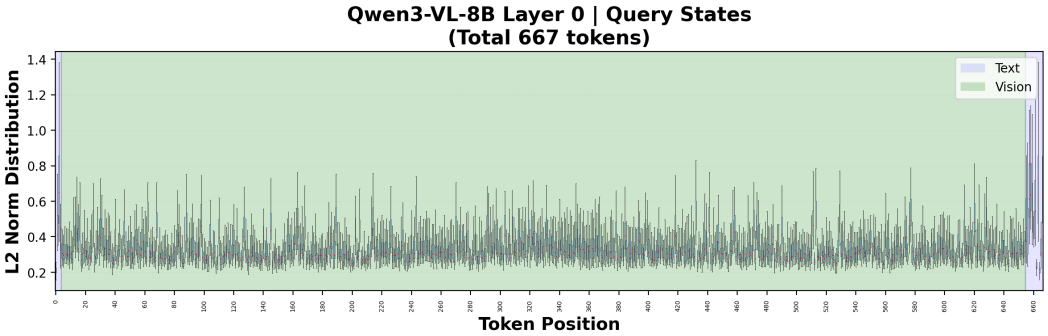


(b) Key L2 norm distribution

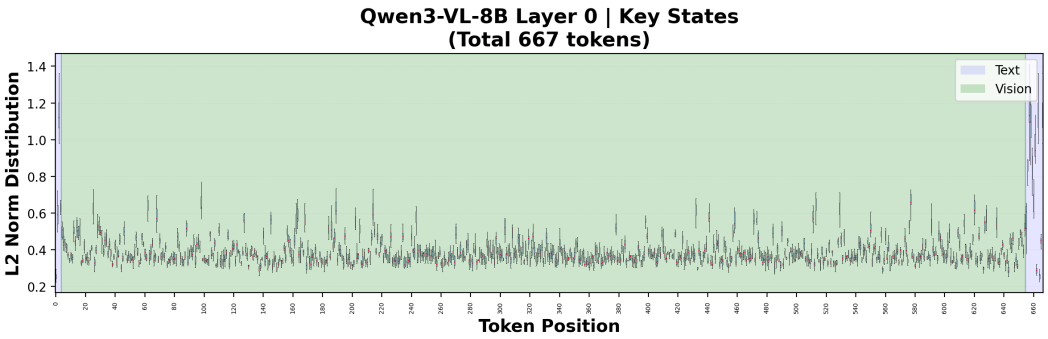


(c) Value L2 norm distribution

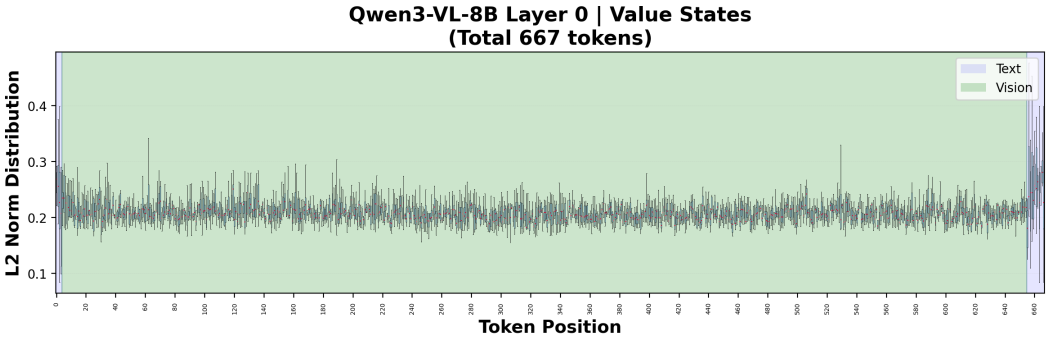
Figure 19: L2 norm distributions of Query, Key, and Value states in Layer 24 of Qwen-3-VL-8B, showing broader token norm variation compared to text-only LLMs.



(a) Query L2 norm distribution

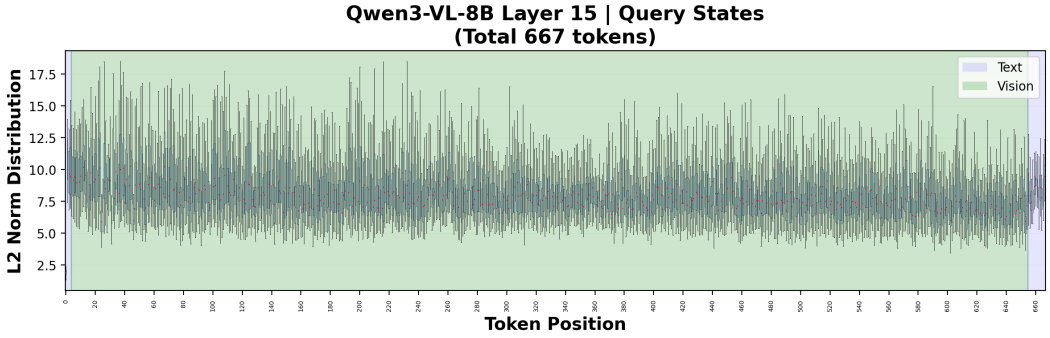


(b) Key L2 norm distribution

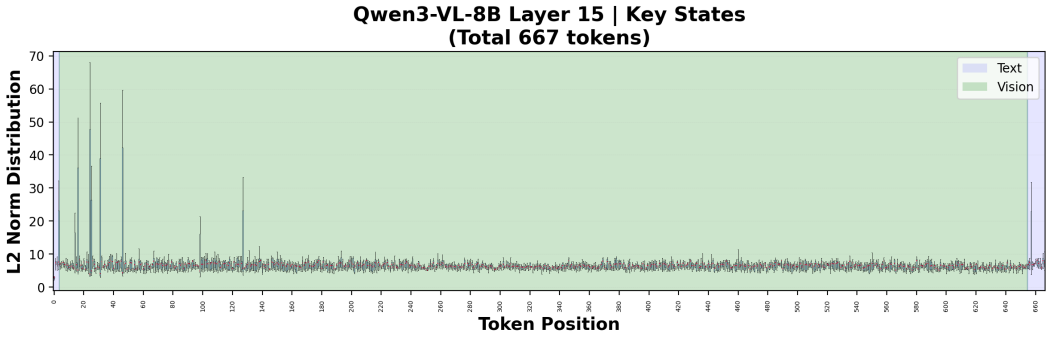


(c) Value L2 norm distribution

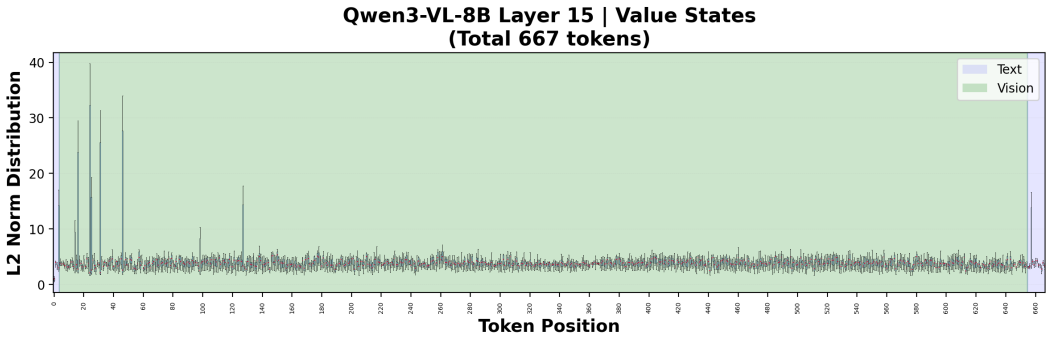
Figure 20: L2 norm distributions of Query, Key, and Value states in Layer 0 of Qwen-3-VL-8B, revealing significant inter-modality norm disparities.



(a) Query L2 norm distribution



(b) Key L2 norm distribution



(c) Value L2 norm distribution

Figure 21: L2 norm distributions of Query, Key, and Value states in Layer 15 of Qwen-3-VL-8B, revealing outlier tokens with exceptionally large norms.

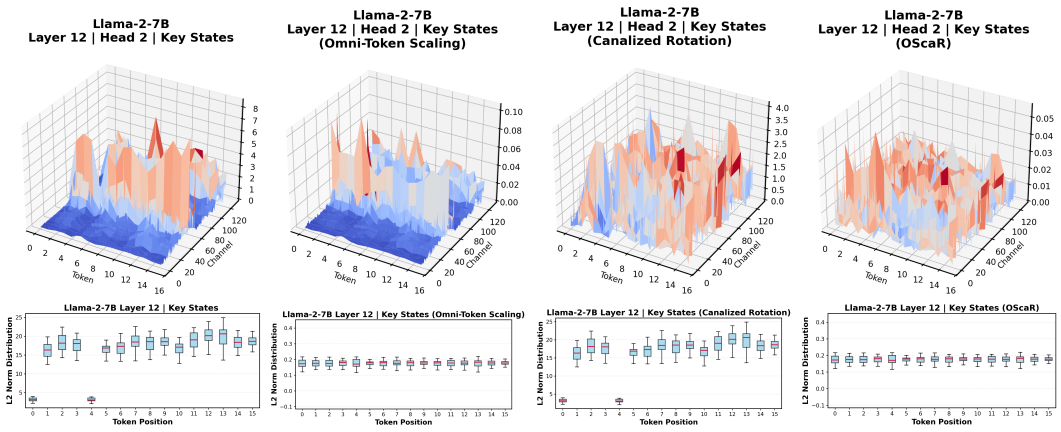


Figure 22: Key magnitude (top row) and L2 norm distribution (bottom row) across different processing stages. Results are shown for Llama-2-7B, Layer 12.

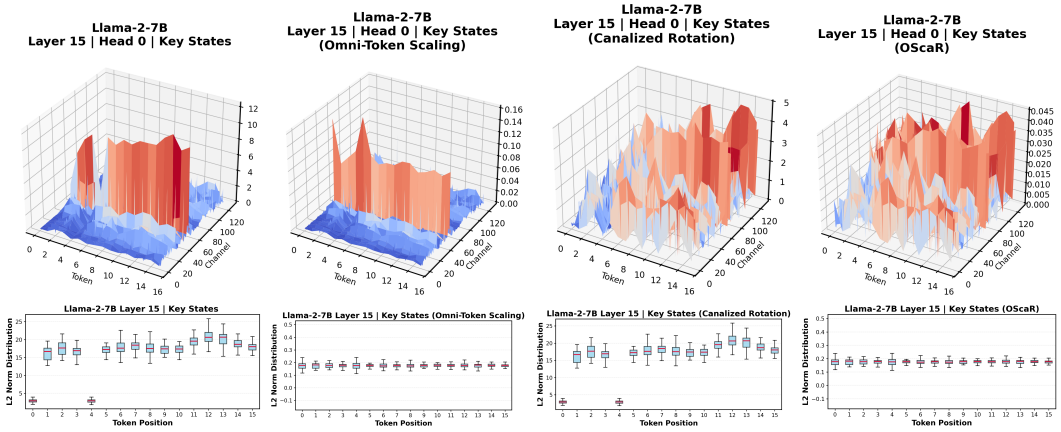


Figure 23: Key magnitude (top row) and L2 norm distribution (bottom row) across different processing stages. Results are shown for Llama-2-7B, Layer 18.

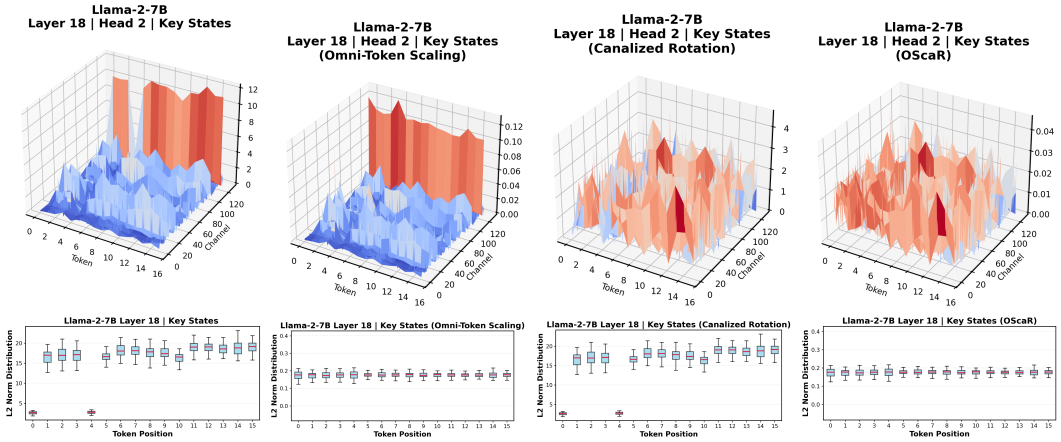
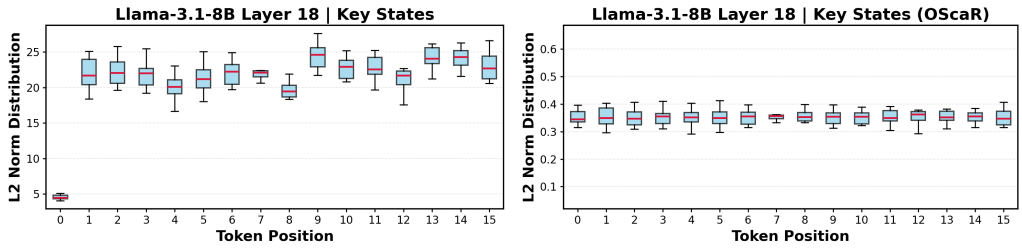
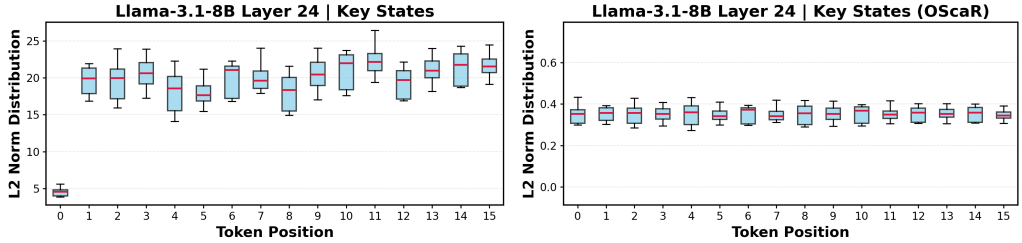


Figure 24: Key magnitude (top row) and L2 norm distribution (bottom row) across different processing stages. Results are shown for Llama-2-7B, Layer 18.



(a) Llama-3.1-8B Layer 18 (before OScaR).

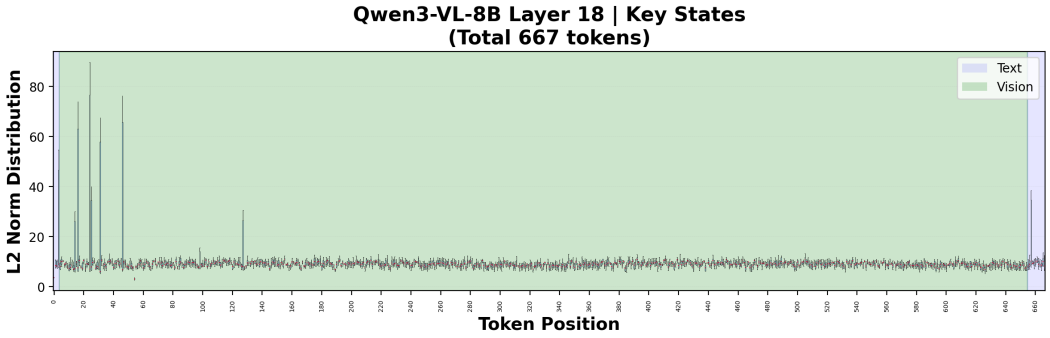
(b) Llama-3.1-8B Layer 18 (after OScaR).



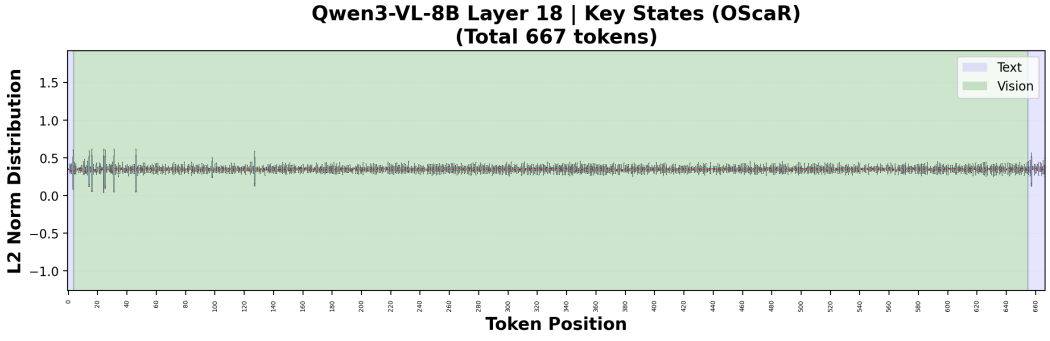
(c) Llama-3.1-8B Layer 24 (before OScaR).

(d) Llama-3.1-8B Layer 24 (after OScaR).

Figure 25: Token norm distribution on Llama-3.1-8B before and after applying OScaR.

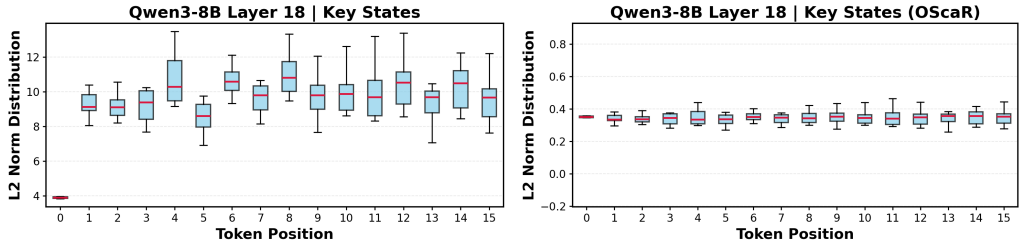


(a) Qwen3-VL-8B (before OScaR).



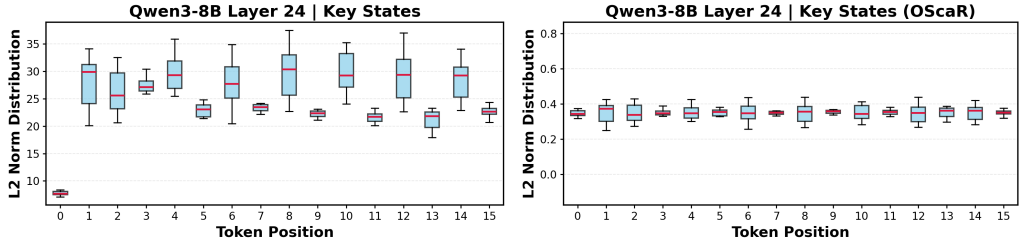
(b) Qwen3-VL-8B (after OScaR).

Figure 26: Token norm distribution on Qwen3-VL-8B before and after applying OScaR.



(a) Qwen3-8B Layer 18 (before OScaR).

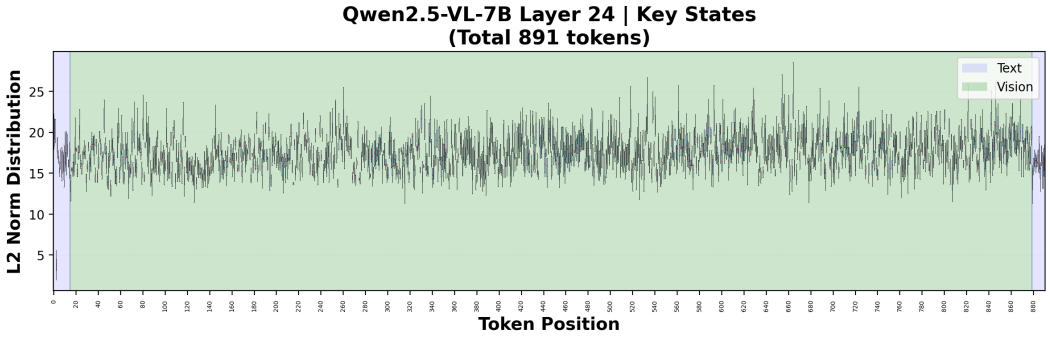
(b) Qwen3-8B Layer 18 (after OScaR).



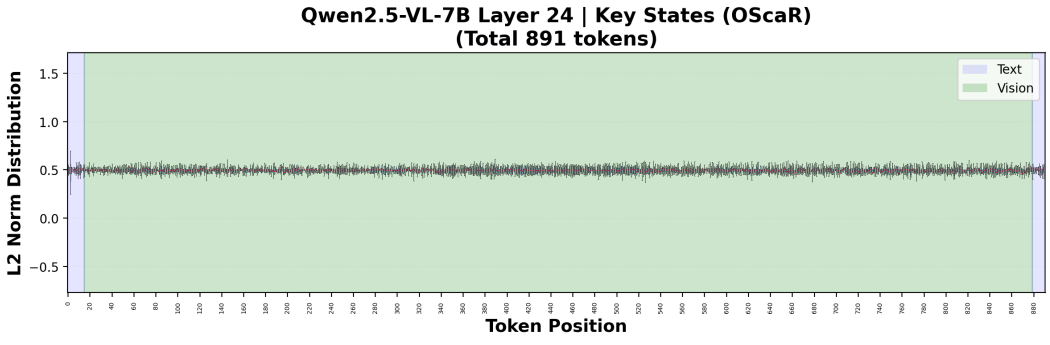
(c) Qwen3-8B Layer 24 (before OScaR).

(d) Qwen3-8B Layer 24 (after OScaR).

Figure 27: Token norm distribution on Qwen3-8B before and after applying OScaR.

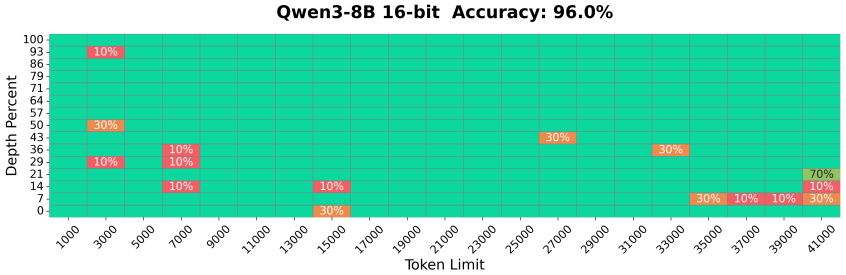


(a) Qwen2.5-VL-7B (before OScaR).

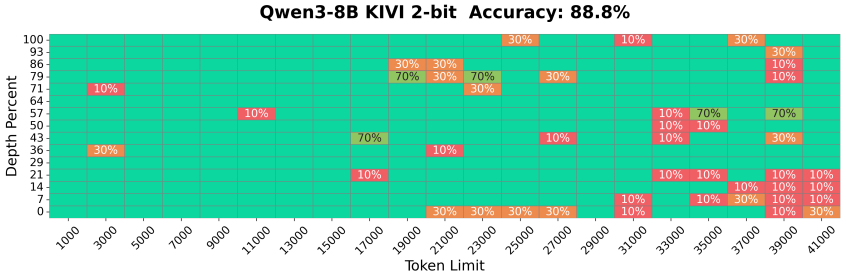


(b) Qwen2.5-VL-7B (after OScaR).

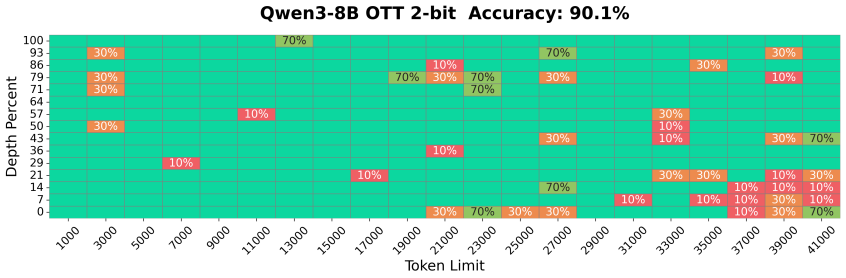
Figure 28: Token norm distribution on Qwen2.5-VL-7B before and after applying OScaR.



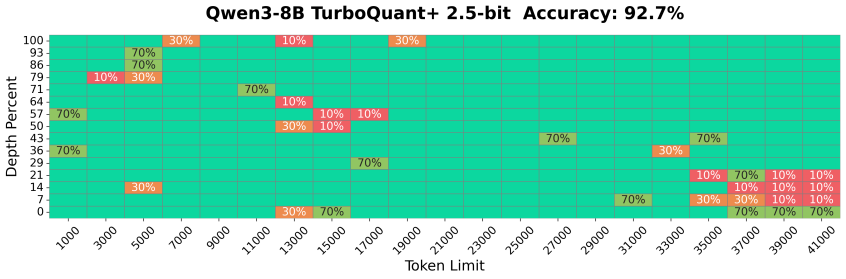
(a) Full-precision baseline with 16-bit KV cache. Retrieval accuracy: 96.0%.



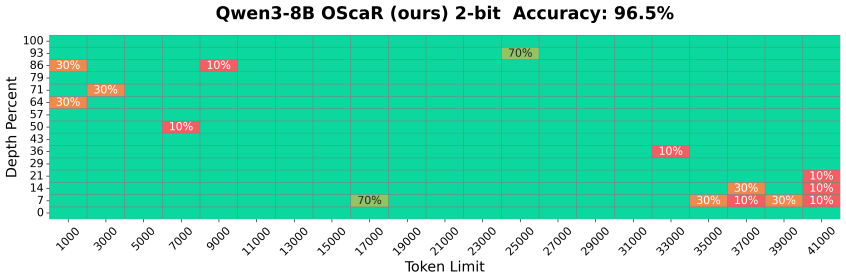
(b) KIVI under 2-bit KV cache quantization. Retrieval accuracy: 88.8%.



(c) OTT under 2-bit KV cache quantization. Retrieval accuracy: 90.1%.



(d) TurboQuant+ under 2.5-bit KV cache quantization. Retrieval accuracy: 92.7%.



(e) OScaR under 2-bit KV cache quantization (our method). Retrieval accuracy: 96.5%.

Figure 29: NIAH evaluation results. All competing methods except TurboQuant+ are configured with INT2 quantization and a group size of 32. TurboQuant+ uses a 2.5-bit setting. TurboQuant is based on TurboQuant+ (Turney & Contributors, 2026); QJL is excluded as it degrades performance. See Appendix O for details.