VQuant: Vertical-Line-Aware Mixed-Precision Quantization

Anonymous ACL submission

Abstract

Quantization has been widely adopted in LLM training and inference pipelines, delivering substantial gains in both cost and efficiency. However, existing low-bit quantization for the *attention* module often introduces large quantization errors at very low bit-widths, leading to noticeable performance degradation . Prior work has mostly focused on smoothing techniques to mitigate outliers, whereas we emphasize a *mixed-precision* design. Motivated by extensive attention heatmap analyses, we observe that LLM attention patterns typically contain a *very small set of dominant vertical lines* that carry a disproportionate amount of attention mass. We preserve this small but crucial subset in full precision during quantized computation. Specifically, we propose **VQuant**. In the **Prefill** stage, we design a new quantized attention operator: during computation, we keep the identified vertical-line positions and a local sliding window in full precision, while quantizing the remaining parts to low bit-width. In the **Decode** stage, we follow the same principle: when quantizing the KV cache, we keep the vertical lines and the local window unquantized, and further fuse KV dequantization with attention computation to improve hardware efficiency. Experiments show that in the **Prefill** stage, VQuant reduces MSE by about **5**× with little extra computation overhead. In the **Decode** stage, VQuant combines KV quantization with fused attention, preserving end-to-end quality across benchmarks while achieving up to **3.58**× speedup. With the two-stage co-design, VQuant achieves near lossless quality in end-to-end evaluations.

1 Introduction

As OpenAI's GPT series (OpenAI, 2024), Gemini (Gemini Team, 2025), and DeepSeek (DeepSeek-AI, 2024) establish Chain-of-Thought (CoT) as a mainstream reasoning paradigm, Large Language Models (LLMs) have shifted from producing brief conclusions to generating long multi-step reasoning traces, often spanning tens of thousands of tokens.

In long-context inference, bottlenecks differ across stages. During *Prefill*, attention over the full prefix is dominated by large matrix multiplications, making latency mainly compute-bound. During *Decode*, each new token repeatedly reads/writes past KV states, causing GPU memory capacity and bandwidth to become the primary constraints.

Quantization is a practical way to reduce both compute and memory costs in LLM inference and is widely used in deployment. However, **uniformly pushing precision to very low bit-widths is often infeasible**: When all tokens share the same bit-width, errors can be disproportionately amplified in long-context reasoning, causing visible end-to-end degradation.(Zheng et al., 2025) This mainly stems from two factors: *(i) non-uniform error sensitivity across positions*—when attention is highly imbalanced, a few high-contribution positions become extremely sensitive, and low-bit noise there can dominate the output error; *(ii) a uniform bit-width mismatches structured token-importance disparity*—token importance spans orders of magnitude, so uniform compression over-compresses critical tokens while wasting budget on insensitive ones. Therefore, a key challenge is to **constrain dominant quantization error sources** and make low-bit quantization reliable for long-context inference.

Prior work on long-context efficiency broadly falls into **sparsification** and **quantization**. **Sparsification** skips low-contribution attention/cache content to save compute or storage. For *Prefill (P)*, block-sparse attention computes only selected blocks, but often requires non-trivial preprocessing (e.g., online scoring, retrieval, reordering) and becomes beneficial mainly at very long contexts; for short/moderate sequences, preprocessing can offset saved FLOPs. For *Decode (D)*, KV management via eviction or selective retention reduces KV foot-

print, e.g., H2O (Zhang et al., 2023), SnapKV (Li et al., 2024), PyramidKV (Cai et al., 2024), and HeadKV (Fu et al., 2024). However, eviction can be brittle for long CoT where token importance is non-stationary: evicted content may later become critical, degrading reasoning quality.

Quantization reduces bandwidth and storage via low-bit representations and can partially lower operator costs. In P, quantized/approximated attention (e.g., SageAttn (Zhang et al., 2025)) often suffers quality loss at more aggressive bit-widths. In decode, KV-cache quantization is relatively mature (e.g., KIVI (Liu et al., 2024), KVTuner (Li et al., 2025), KVQuant (Hooper et al., 2024)), where smoothing/outlier suppression mitigates error but typically treats tokens as homogeneous, overlooking the structured imbalance of token sensitivity.

Orthogonally, system optimizations such as FlashAttention (Dao et al., 2022), PagedAttention (Kwon et al., 2023), and Mooncake (Qin et al., 2024) improve IO efficiency and KV management. Yet they mainly address scheduling, layout, and orchestration, and **do not explicitly answer a reasoning-centric mixed-precision question**: How to design a mixed-precision mechanism that accounts for heterogeneous token sensitivity to balance accuracy and efficiency in long-context CoT inference.

As shown in Fig. 1, we repeatedly observe salient *vertical-line* structures across many attention heatmaps, consistent with prior observations (Jiang et al., 2024). Compared to diagonal/oblique bands that drift with generation, these vertical lines carry a higher and more stable attention mass, but cover only a tiny fraction of tokens (typically $< 5\%$). This suggests preserving such positions with full precision during low-bit quantization to suppress errors dominated by a few high-contribution tokens.

Motivated by observations, we propose **VQuant**, a **mixed-precision quantization** method. To avoid catastrophic errors under uniform low-bit settings, **VQuant** keeps full-precision for a tiny set of error-sensitive positions: (1) persistent *vertical-line* positions, and (2) a recent *sliding window* for short-range dependencies. Unlike sparsification methods with heavy preprocessing or expensive online selection, VQuant identifies vertical lines with a lightweight procedure and uses a pre-specified window size, incurring negligible overhead.

Prefill (compute-intensive): quantized attention for speed. We design a quantized attention operator that executes most prefill computation in low precision while keeping vertical lines and the local window in FP to avoid error amplification; we further *fuse* the low-precision and FP paths into a single operator with minimal overhead over naive low-bit attention.

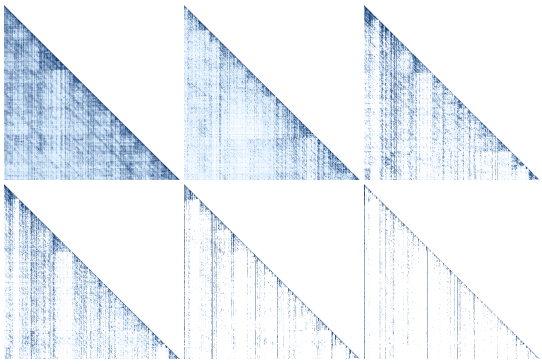

Figure 1: **Vertical-line-dominated attention in real inference (Qwen3-8B).** Across multiple layers, persistent vertical bright stripes indicate that a small number of key positions are repeatedly attended by many queries over long spans, revealing a highly imbalanced attention distribution and motivating compression strategies that prioritize a small set of critical positions.

Decode (memory-intensive): low-bit KV cache for footprint. We quantize KV cache to low bits to reduce memory capacity and bandwidth pressure, while keeping the same small set of critical positions (vertical lines and the local window) unquantized; we additionally fuse KV dequantization with attention computation to improve hardware efficiency.

Overall, **VQuant** is a *quantization-first* approach: It accelerates compute-bound prefill via quantized attention and enables bandwidth-bound decode via low-bit KV cache, while preserving a minimal FP token budget to suppress dominant quantization errors and maintain quality under aggressive compression. We evaluate **VQuant** on Qwen3-8b with various datasets, including longbench, GSM8K, and math500, comprehensively evaluating its math reasoning ability and long-context task ability. Results show that our method achieves 1.32x ~3.58x acceleration (increasing with the sequence length), meanwhile outperforms other methods in accuracy, aligning with the baseline model.

2 Motivation

2.1 Vertical-Line-Dominated Attention in Real Inference

VQuant is motivated by a recurring observation in real inference: LLM attention maps are highly structured rather than approximately uniform. In long chain-of-thought (CoT) trajectories, we often observe a small number of prominent *vertical lines*—i.e., a tiny subset of key/value positions that continuously attract substantial attention mass from many queries over long spans. These positions typically correspond to *anchor* information (e.g., key entities, constraints, intermediate conclusions, or repeatedly cited evidence), and are revisited throughout subsequent reasoning steps.

More broadly, attention heatmaps in autoregressive models often exhibit a "vertical–diagonal" pattern: The *diagonal band* reflects local dependence on recent context, while *vertical lines* indicate a few fixed positions that remain salient across steps. In long-sequence reasoning, vertical lines usually carry more concentrated attention mass; in contrast, the diagonal band shifts with generation and plays a more transient routing role. Therefore, inference-time KV-cache management should prioritize these cross-step persistent vertical anchors.

To illustrate this phenomenon, Fig. 1 shows attention heatmaps from **Qwen3-8B** at several representative layers, where rows/columns correspond to query/key positions. Beyond the local neighborhood structure, we can clearly observe vertical bright stripes spanning many rows: A small set of key positions contributes a disproportionate amount of attention mass to a large number of queries. This suggests that the *effective attention support* in long-context inference can be far smaller than the sequence length itself. Consequently, treating all KV positions as homogeneous—and applying uniform eviction or uniform low-bit quantization—can be mismatched with real attention dynamics.

Evidence: a local window and a small set of high-score positions cover most attention mass. We further quantify this "vertical-line-dominant" phenomenon on **Qwen3-8B** across different context lengths L. We measure the fraction of attention mass (softmax$(QK^\top)$) covered by: (i) a local window ($W{=}128$, WIN), (ii) the global top-1% highest-scoring key positions (TOP-1%), and (iii) their union (TOTAL). As shown in Table 1, when L increases from 4K to 32K, WIN $\cup$ TOP-1% consistently covers more than half of the total attention mass (about $53.24\% \sim 59.60\%$), indicating that the effective attention support remains highly concentrated and further motivating high-fidelity preservation for a small set of critical positions.

Table 1: **Attention-mass coverage across context lengths (Qwen3-8B).** We report the fraction of attention mass (softmax$(QK^\top)$) covered by the local window $W{=}128$ (WIN), the global top-1% key positions (TOP-1%), and their union (TOTAL).

Length L	4,096	8,192	16,384	32,768
WIN	29.21%	21.25%	19.60%	16.81%
TOP-1%	30.39%	33.60%	35.26%	36.43%
TOTAL	59.60%	54.85%	54.86%	53.24%

These results suggest that KV compression should be *non-uniform*: we preserve a tiny set of high-attention vertical anchors in high precision while aggressively compressing the rest, which is the key idea behind **VQuant**.

2.2 Quantization Errors Are Amplified at High-Score Positions

Low-bit quantization errors are not equally harmful across tokens and are most destructive at *high-score* positions. When attention concentrates on a few vertical-line tokens, the attention output becomes highly sensitive to perturbations at these locations: Their softmax weights are larger, and they are repeatedly accessed by many queries, so the same K/V quantization noise is more likely to be amplified through KV reuse across prefill and decode. Therefore, uniform quantization with a fixed compression rate often fails to protect a small number of critical tokens, leading to considerable quality degradation. In particular, loss on *vertical-line anchors* directly harms cross-step evidence re-reading and the stability of long-range reasoning.

Evidence: protecting a tiny set of high-score positions substantially reduces error. We conduct error analysis on **Qwen3-8B** and use full-precision FlashAttention as the teacher reference. We compare quantization strategies by measuring the mean squared error (MSE) of the intermediate attention output (the input to o_proj). As shown in Fig. 4, uniform 4-bit quantization yields noticeably larger errors than uniform 8-bit quantization. Crucially, under the same 4-bit budget, keeping only a small set of *high-score vertical-line positions* in full pre-

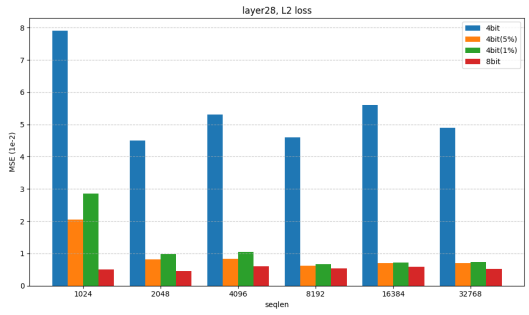

Figure 2: **Amplified quantization error at high-score positions (Qwen3-8B, Layer 28).** Uniform 4-bit quantization incurs substantially larger MSE than uniform 8-bit quantization. Under the same 4-bit budget, keeping only the top-1%/top-5% high-score vertical-line positions in full precision significantly reduces the error and consistently approaches the 8-bit error level across different sequence lengths.

cision dramatically suppresses the error. For example, at 32K, protecting the top-1% positions reduces the MSE of Layer 28 by about 5.8× relative to uniform 4-bit quantization, bringing it much closer to uniform 8-bit quantization. This trend holds consistently across sequence lengths (1K–32K), suggesting that the overall quantization error is often dominated by a tiny fraction of high-score positions. This directly motivates our mixed-precision design: with a small full-precision budget, we prioritize protecting critical vertical-line K/V while quantizing the remaining majority to low bit-width.

3 Related Work

Long-context sparsification. A major line of work accelerates long-context inference by skipping low-contribution attention or cache content. For *Prefill*, block-sparse attention reduces FLOPs by computing only selected blocks, often relying on online scoring, retrieval, or reordering, and tends to pay off mainly at very long contexts when preprocessing overhead is amortized. For *Decode*, KV-cache eviction or selective retention reduces memory footprint, including H2O (Zhang et al., 2023), SnapKV (Li et al., 2024), PyramidKV (Cai et al., 2024), and HeadKV (Fu et al., 2024). However, eviction can be brittle for long CoT where token importance is non-stationary, since removed content may later become critical.

Attention and KV-cache quantization. Quantization reduces bandwidth and storage via low-bit representations. In *Prefill*, quantized/approximated attention (e.g., SageAttention) can suffer noticeable quality loss under aggressive bit-widths (Zhang et al., 2025). In *Decode*, KV-cache quantization is more mature, including KIVI (Liu et al., 2024), KVTuner (Li et al., 2025), and KVQuant (Hooper et al., 2024), which mitigate error via smoothing or outlier suppression. Most of these methods, however, treat tokens as largely homogeneous, and do not explicitly incorporate the structured imbalance of token sensitivity that becomes pronounced in long-context reasoning (Zheng et al., 2025).

4 Preliminaries

4.1 FlashAttention

Standard attention is

$$O = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V. \tag{1}$$

For long sequences, the implementation involves large intermediate tensors and incurs heavy memory traffic. FlashAttention(Dao, 2024) adopts a block-wise schedule: for each Q block, it iteratively scans K/V blocks, and maintains numerically stable normalization and accumulation via online softmax inside the kernel.

Specifically, when scanning the j-th K/V block, let m and l denote the running maximum and running normalizer. For scores $s^{(j)}$, we update

$$m^{(j)} = \max\left(m^{(j-1)},\ \max_i s_i^{(j)}\right). \tag{2}$$

$$l^{(j)} = l^{(j-1)}\exp\left(m^{(j-1)} - m^{(j)}\right) + \sum_i \exp\left(s_i^{(j)} - m^{(j)}\right). \tag{3}$$

The numerator term is accumulated in the same stable manner, and the final output is $O = o^{(J)}/l^{(J)}$. In the decode stage (single-step $|Q|{=}1$), FlashDecode uses split-K parallelism to scan a long prefix and combines block-wise statistics via a stable reduction, improving parallelism and reducing per-step latency.

5 Method

We present **VQuant**, a vertical-line-aware mixed-precision quantization method for long-context inference. As illustrated in Fig. 1, a tiny fraction of tokens (typically $< 5\%$) forms persistent *vertical lines* that carry disproportionately large attention mass and dominate quantization error under aggressive low-bit settings. VQuant explicitly preserves

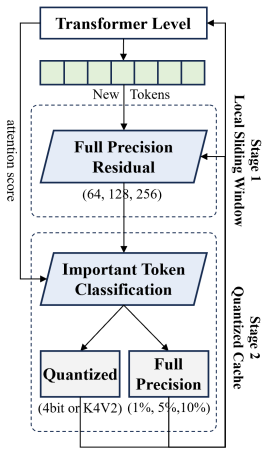

Figure 3: **Flowchart of the Vquant method**

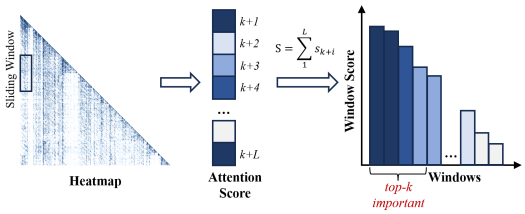

Figure 4: **The vertical line awareness process**

these error-sensitive positions (together with a local window) in full precision, while quantizing the remaining majority to low bits. Concretely, our method consists of three components: **(i) vertical-line awareness**, **(ii) P-stage quantized attention**, and **(iii) D-stage low-bit KV cache with fused decode attention**.

5.1 Vertical-line awareness

Token sets. For each layer/head, we partition the key positions into three disjoint subsets:

$$\mathcal{K} = \mathcal{K}_{\text{VL}} \cup \mathcal{K}_{\text{Win}} \cup \mathcal{K}_{\text{Q}}, \tag{4}$$

where $\mathcal{K}_{\text{VL}}$ denotes the *vertical-line* positions, $\mathcal{K}_{\text{Win}}$ denotes a fixed recent sliding window, and $\mathcal{K}_{\text{Q}}$ denotes the remaining majority to be quantized.

Window definition. Given the current query index t, the window keys are

$$\mathcal{K}_{\text{Win}}(t) = \{\, k \mid k \in [\max(0, t-W+1),\ t] \,\}, \tag{5}$$

where W is a pre-specified window size.

Vertical-line identification (lightweight). We maintain a running importance score for each key position that measures the accumulated *column mass* (vertical attention mass). Let $a_{t,k}$ be the attention probability from query t to key k. We define the vertical-line score as

$$S(k) = \sum_{t \in \mathcal{T}} a_{t,k}, \tag{6}$$

where $\mathcal{T}$ is the set of queries considered (e.g., within the current segment or within a running buffer). We then select the top-ρ fraction as vertical-line positions:

$$\mathcal{K}_{\text{VL}} = \text{TopK}\big(S,\ \rho \cdot |\mathcal{K}|\big), \qquad \rho \ll 1. \tag{7}$$

In practice, ρ is small. Importantly, $S(k)$ can be updated with negligible overhead using lightweight reduction, and the window size W is fixed in advance; therefore, **vertical-line awareness introduces virtually no extra preprocessing cost** compared to sparsification pipelines.

5.2 Prefill-stage quantized attention

Mixed-precision attention in Prefill. In Prefill, attention is compute-intensive and dominated by large GEMMs. We quantize the majority of attention computation, while preserving full precision on $\mathcal{K}_{\text{VL}}$ and $\mathcal{K}_{\text{Win}}$.

Specifically, for a query block Q, we conceptually split keys/values into

$$(K, V) = (K_{\text{FP}}, V_{\text{FP}}) \cup (K_{\text{Q}}, V_{\text{Q}}), \tag{8}$$

where $(K_{\text{FP}}, V_{\text{FP}})$ correspond to $\mathcal{K}_{\text{VL}} \cup \mathcal{K}_{\text{Win}}$, and $(K_{\text{Q}}, V_{\text{Q}})$ correspond to $\mathcal{K}_{\text{Q}}$.

We compute attention with a fused operator:

$$O = \text{Softmax}\left(\frac{Q K_{\text{FP}}^{\top}}{\sqrt{d}} \,\|\, \frac{Q \hat{K}_{\text{Q}}^{\top}}{\sqrt{d}} \right) \cdot \left(V_{\text{FP}} \,\|\, \hat{V}_{\text{Q}} \right), \tag{9}$$

where $\hat{K}_{\text{Q}}, \hat{V}_{\text{Q}}$ are dequantized values of low-bit $(K_{\text{Q}}, V_{\text{Q}})$, and $\|$ denotes concatenation along the key dimension. Crucially, we *fuse* the FP path and the quantized path into a single kernel so that the operator structure remains FlashAttention-like (online softmax with blockwise scanning), and the additional overhead over naive low-bit attention quantization is minimal.

Implementation note. In our implementation, we use lower precision for the bulk computation (e.g., INT8/FP8, INT4/FP4 depending on the backend), while keeping the vertical-line and window tiles in FP16/BF16. This mixed execution preserves the accuracy-critical mass with a tiny FP budget.

Algorithm 1 VQuant Prefill: Vertical-Line-Aware Mixed-Precision Quantized Attention (Fused Kernel)

Input: query block Q_b; keys/values K, V; window size W; vertical ratio ρ; bitwidth B; tile sizes (B_M, B_N); scale $\alpha = 1/\sqrt{d}$
Output: output block O_b (causal)

determine protected keys: vertical lines + local window
$\mathcal{K}_{\text{WIN}} \leftarrow \text{LocalWin}(b, W)$ ▷ keys within the recent window of this block
update vertical scores $S(k)$ by lightweight reduction ▷ Eq. (6)
$\mathcal{K}_{\text{VL}} \leftarrow \text{TopK}(S, \lceil \rho|\mathcal{K}| \rceil)$ ▷ Eq. (7)
$\mathcal{K}_{\text{FP}} \leftarrow \mathcal{K}_{\text{VL}} \cup \mathcal{K}_{\text{WIN}}$; $\mathcal{K}_{\text{Q}} \leftarrow \mathcal{K} \setminus \mathcal{K}_{\text{FP}}$
FlashAttention-style online softmax state
$(m, \ell, acc) \leftarrow (-\infty, 0, \mathbf{0})$ ▷ $m, \ell \in R$, $acc \in R^{B_M \times d}$
# fused scan over KV tiles (single kernel)
for all each key tile T of size B_N (causal) **do do**
  $\mathcal{I} \leftarrow \text{Indices}(T)$
  $\mathcal{I}_{\text{FP}} \leftarrow \mathcal{I} \cap \mathcal{K}_{\text{FP}}$; $\mathcal{I}_{\text{Q}} \leftarrow \mathcal{I} \cap \mathcal{K}_{\text{Q}}$
  if $\mathcal{I}_{\text{FP}} \neq \emptyset$ **then**
    $k \leftarrow K[\mathcal{I}_{\text{FP}}]$; $v \leftarrow V[\mathcal{I}_{\text{FP}}]$
    $(m, \ell, acc) \leftarrow (Q_b, k, v; \alpha, m, \ell, acc)$
  end if
  if $\mathcal{I}_{\text{Q}} \neq \emptyset$ **then**
    $k^{\text{Q}} \leftarrow_B (K[\mathcal{I}_{\text{Q}}])$; $v^{\text{Q}} \leftarrow_B (V[\mathcal{I}_{\text{Q}}])$ ▷ packed B-bit
    $(\hat{k}, \hat{v}) \leftarrow (k^{\text{Q}}, v^{\text{Q}})$ ▷ in-kernel, no materialization
    $(m, \ell, acc) \leftarrow (Q_b, \hat{k}, \hat{v}; \alpha, m, \ell, acc)$
  end if
end for
$O_b \leftarrow acc/\ell$
return O_b

Algorithm 2 VQuant Decode: Vertical-Line-Aware Mixed-Precision KV with Fused Dequant-Attention

Input: q_t (single-token query at step t); mixed KV cache $\{K^{\text{FP}}, V^{\text{FP}}, K^{\text{Q}}, V^{\text{Q}}\}$; vertical ratio ρ; local window W; group size G; bitwidth B; tile size B_N; split size S; scale $\alpha = 1/\sqrt{d}$
Output: o_t

protected keys (vertical lines + local window)
$\mathcal{K}_{\text{WIN}} \leftarrow [\max(0, t - W + 1),\ t]$
$\mathcal{K}_{\text{VL}} \leftarrow \text{TopK}(S, \lceil \rho(t+1) \rceil)$ ▷ running vertical score $S(k)$
$\mathcal{K}_{\text{FP}} \leftarrow \mathcal{K}_{\text{VL}} \cup \mathcal{K}_{\text{WIN}}$
$\mathcal{K}_{\text{Q}} \leftarrow [0, t] \setminus \mathcal{K}_{\text{FP}}$
initialize online softmax state
$(m, \ell, acc) \leftarrow (-\infty, 0, \mathbf{0})$ ▷ $m, \ell \in R$, $acc \in R^d$
# single-pass scan over prefix KV tiles (causal)
for $T \leftarrow 0$ **to** $\lceil (t+1)/B_N \rceil - 1$ **do**
  $\mathcal{I} \leftarrow [T \cdot B_N,\ \min((T+1)B_N - 1,\ t)]$
  $\mathcal{I}_{\text{FP}} \leftarrow \mathcal{I} \cap \mathcal{K}_{\text{FP}}$; $\mathcal{I}_{\text{Q}} \leftarrow \mathcal{I} \cap \mathcal{K}_{\text{Q}}$
  if $\mathcal{I}_{\text{FP}} \neq \emptyset$ **then**
    $k \leftarrow K^{\text{FP}}[\mathcal{I}_{\text{FP}}]$; $v \leftarrow V^{\text{FP}}[\mathcal{I}_{\text{FP}}]$
    $(m, \ell, acc) \leftarrow (q_t, k, v; \alpha, m, \ell, acc)$
  end if
  if $\mathcal{I}_{\text{Q}} \neq \emptyset$ **then**
    Split $\mathcal{I}_{\text{Q}}$ into S chunks and compute per-chunk states in parallel
    $(m, \ell, acc) \leftarrow (\{m_s, \ell_s, acc_s\}_{s=1}^{S})$ ▷ stable merge as in FlashDecode
  end if
end for
$o_t \leftarrow acc/\ell$
return o_t

5.3 Decode-stage low-bit KV cache and fused attention

Low-bit KV cache with selective full precision. Decode is memory-intensive: each step reads a large prefix KV. We store KV in a mixed manner:

$$(K_{1:t}, V_{1:t}) = (K^{\text{FP}}_{1:t}, V^{\text{FP}}_{1:t}) \cup (K^{\text{Q}}_{1:t}, V^{\text{Q}}_{1:t}), \quad (10)$$

where full precision is reserved for keys in $\mathcal{K}_{\text{VL}} \cup \mathcal{K}_{\text{Win}}(t)$, and the remaining majority is stored in low bits. This directly reduces KV memory footprint and bandwidth pressure.

Fused dequantization + attention. To avoid extra memory traffic, we fuse dequantization into the decode attention kernel: when scanning a quantized KV block, we dequantize on-the-fly and immediately apply it to score/value accumulation (FlashAttention-style online softmax). Formally, for a quantized block $(K^{\text{Q}}_b, V^{\text{Q}}_b)$,

$$\hat{K}_b = \text{DeQuant}(K^{\text{Q}}_b), \hat{V}_b = \text{DeQuant}(V^{\text{Q}}_b) \quad (11)$$

and the kernel updates the running softmax state and output accumulator using $(\hat{K}_b, \hat{V}_b)$ without materializing full-precision intermediates. For blocks belonging to $\mathcal{K}_{\text{VL}} \cup \mathcal{K}_{\text{Win}}(t)$, we directly load FP keys/values.

Summary. By keeping only a tiny set of vertical-line positions and a local window in full precision, VQuant controls the dominant quantization errors, while quantizing the remaining majority to achieve low-bit efficiency. Prefill benefits from quantized attention speedups, and decode benefits from reduced KV memory together with fused dequantization for bandwidth efficiency.

6 Experiments

6.1 Experimental Setup

Model. We evaluate on a representative LLM, **Qwen3-8B** (Yang et al., 2025). We focus on its *long-context reasoning* capability, and adopt predominantly *chain-of-thought* task settings to cover multi-step reasoning and information aggregation under long sequences.

Baselines. We compare against three representative methods: **(i) Full-precision baseline: FlashAttention-2** (Dao, 2024), which serves as the accuracy reference and latency baseline; **(ii)**

KIVI (Liu et al., 2024), which compresses the KV cache with low-bit quantization to reduce memory/bandwidth overhead during long-context inference while preserving accuracy as much as possible; **(iii) SageAttention** (Zhang et al., 2025), which accelerates attention by mitigating activation outliers and using a numerically stable computation path to maintain inference accuracy.

Implementation Details (VQuant). Unless specified, all experiments are conducted on an **NVIDIA A100** GPU. VQuant adopts a mixed-precision KV design for both the Prefill and Decode stages. We keep two categories of positions in full precision: (i) the online-identified top-5% vertical-line tokens, and (ii) tokens inside a local sliding window. All remaining KV positions are stored in Key-4bit, Value-2bit quantized formats to ensure that the memory usage does not exceed normal 4bit quantization. Quantized caches are recovered via dequantization and mixed-precision computation to balance accuracy and efficiency.

Benchmarks and metrics. For accuracy, we evaluate mathematical reasoning on **GSM8K** (Cobbe et al., 2021) and **MATH500** (Hendrycks et al., 2021), and assess long-context capability on **LongBench v1** (Bai et al., 2024). For efficiency and operator-level analysis, we focus on the Prefill and Decode stages and report: (i) **Prefill operator-level error**: using full-precision FlashAttention-2 as the reference, we compute the layer-wise MSE loss of the intermediate attention output (i.e., the input to o_{proj}) to quantify numerical deviations introduced by low-bit computation, and compare against **SageAttention**; (ii) **Decode latency**: under the same decoding setup, we report the per-step total decoding latency (ms/token) across different prefix lengths and compare with the **FlashAttention** baseline, together with the relative speedup. All latency numbers are collected after sufficient warmup and averaged over multiple runs.

6.2 Main Results

6.2.1 Benchmark Results

Tables 2 and 3 summarize end-to-end benchmark results. On **Qwen**, VQuant matches the performance of full-precision attention (FlashAttention-2), while consistently outperforming KIVI and SageAttention under the same or lower KV storage budget. This indicates that a mixed-precision strategy that *preserves a small set of critical tokens together with a local window* is sufficient to retain long-context understanding and math reasoning ability. For Qwen, we **enable thinking mode** in end-to-end evaluation to better reflect realistic long-CoT generation and KV reuse.

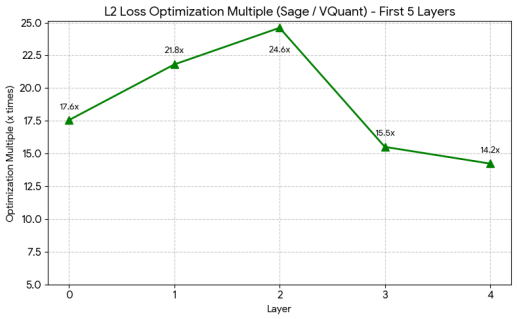

Figure 5: **Layer-wise MSE optimization multiple in Prefill (first 5 layers).** Using FlashAttention-2 as the reference, we compute the MSE of the intermediate attention output (input to o_{proj}) per layer. We plot the optimization multiple $\text{MSE}_{\text{Sage}}/\text{MSE}_{\text{VQuant}}$, where larger is better.

6.2.2 Operator-level evaluation: Prefill error and Decode latency

Prefill: layer-wise MSE. Since our Prefill latency is nearly identical to SageAttention, we focus on numerical deviations in this stage. We use full-precision FlashAttention-2 as the reference and compute the **MSE** of the intermediate attention output (i.e., the input to o_{proj}) for each layer. We further report the **optimization multiple** of VQuant over SageAttention, defined as $\text{MSE}_{\text{Sage}}/\text{MSE}_{\text{VQuant}}$. As shown in Fig. 5, we visualize the first five layers as representative examples. VQuant achieves a large reduction in MSE over SageAttention in these layers, demonstrating that *selectively preserving vertical-line tokens and the local window in full precision* effectively suppresses error amplification under low-bit computation.

Decode: decoding latency vs. FlashAttention. In the Decode stage, we compare VQuant against FlashAttention in terms of per-step total decoding latency (ms/token) under different prefix lengths (seq_len). Table 4 summarizes the results: as the context grows, VQuant achieves increasingly larger gains, reaching about $3.58\times$ speedup over FlashAttention at the 32K prefix length. This highlights the sustained benefits of *mixed-precision KV storage and fused dequantization* for bandwidth-/memory-

Table 2: **LongBench v1 results on Qwen3-8B (thinking mode).** We compare full-precision attention (FA2), KIVI, SageAttention, and VQuant (Ours). **Avg** denotes the average score over all tasks.

Method	Single-Document QA			Multi-Document QA			Summarization		Few-shot Learning		Semantic	Avg.
	NarrQA	Qasper	MF-en	HotpotQA	2Wiki	Musique	QMSum	MNews	Trivia	SAMSum	Pretrieve-en	
FA2 (full precision)	30.37	41.68	**50.34**	**50.42**	39.69	33.88	13.11	10.85	**97.50**	**33.80**	**88.81**	49.04
KIVI (K4V4)	24.76	35.13	45.51	42.55	31.04	18.92	10.51	8.19	88.87	27.56	81.41	41.44
SageAttention	19.45	34.54	41.95	46.59	**44.33**	24.31	**18.67**	**23.23**	74.93	32.62	58.86	41.94
Ours (K4V2, top-5%)	**32.28**	**41.82**	49.74	49.75	43.31	**35.24**	15.39	11.94	96.65	33.05	86.32	**49.67**

Table 3: **Math reasoning results on Qwen3-8B (GSM8K / MATH500).** We compare full-precision attention (FA2), KIVI, SageAttention, and VQuant (Ours).

Method	GSM8K	MATH500
FA2 (full precision)	95.88	80.14
KIVI	92.48	72.89
SageAttention	94.99	77.60
Ours (K4V2, top-5%)	96.52	78.73

Table 4: **Decode latency under different prefix lengths (ms/token).** Speedup = FlashAttention / Ours

seq_len	FlashAttn	Ours	Speedup
1,024	0.224	0.170	1.32×
2,048	0.416	0.173	2.40×
4,096	0.805	0.263	3.06×
8,192	1.604	0.478	3.36×
16,384	3.181	0.903	3.52×
32,768	6.354	1.775	3.58×

Table 5: **Ablation on the critical-token ratio k (placeholder).** Acc is the end-to-end accuracy (e.g., GSM8K accuracy), Speedup is the Prefill operator speedup, and MSE is computed against FA2.

k	Acc	MSE($1e^{-2}$)
1%	94.03	11.24
5%	95.52	8.48
10%	96.30	5.81

Table 6: **Ablation on the local full-precision window (placeholder).** The window can be specified by the number of tokens (e.g., 64/128/256).

Window	Acc	MSE($1e^{-2}$)
64	92.54	9.58
128	95.52	8.48
256	95.74	8.21

bound long-context decoding.

Summarization: While testing on Qwen3-8B, our method achieves both high accuracy and speed. Note that on some datasets, our method outperforms a full precision model. This can be explained that our "token classification" strategy compels model to focus on critical tokens, which may trigger an implicit denoising effect.

6.3 Ablations

6.3.1 Critical-token ratio (k)

We vary the preserved critical-token ratio k (e.g., $1\%, 5\%, 10\%$) while keeping other settings fixed, to study the accuracy–efficiency trade-off (Table 5). Results show that increasing the top-k ratio triggers a decrease in MSE and an increase in accuracy, yet with a higher memory cost.

6.3.2 Local full-precision window size/ratio

We vary the local full-precision window size (or ratio) to evaluate sensitivity to *local-context dependency*, and verify its complementarity with the vertical-line token preservation strategy(Table 6). Results show that a longer window size slightly increases accuracy. This can be explained that more tokens remain full precision during inference.

7 Conclusion

In this paper, we present a new KVcache quantization infrastructure — Vquant, integrating attention-based quantization strategy, novel acceleration operators and mixed-precision attention. Specifically, we classify the importance of tokens based on accumulated attention scores along the vertical direction, based on which we assign different quantization accuracy. We also implement new operators that can accelerate the classification process. In addition, we employ quantization and fused attention to further suppress computation costs, accelerating inference speed while maintaining high accuracy. Experiment results show that our method achieves 1.32 ~3.58 times speedup, while retaining accuracy on several datasets compared with current methods.

Limitations

- Our method heavily relies on the vertical attention pattern that appeared in Qwen3-8B. However, in other models, the vertical attention pattern may not be the dominant pattern, limiting the potential of this method.
- The effect of preserving critical tokens can be fully observed in long-context tasks. When dealing with tasks with short prompts and answers, we acknowledge that VQuant may not significantly improve the performance as in experiments described above.
- In our study, we use an NVIDIA A100 GPU. We acknowledge that when using high-power GPUs, VQuant may not significantly improve over the baseline. Also,due to the limitation of GPU size, we did not carry out our tests on large models, such as Qwen3-80B.

References

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

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. 2024. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Tri Dao. 2024. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness.

DeepSeek-AI. 2024. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *Preprint*, arXiv:2405.04434.

Yu Fu, Zefan Cai, Abedelkadir Asi, Wayne Xiong, Yue Dong, and Wen Xiao. 2024. Not all heads matter: A head-level kv cache compression method with integrated retrieval and reasoning.

Gemini Team. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *Preprint*, arXiv:2507.06261.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. In *NeurIPS*.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. 2024. Kvquant: Towards 10 million context length llm inference with kv cache quantization. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention.

Xing Li, Zeyu Xing, Yiming Li, Linping Qu, Hui-Ling Zhen, Wulong Liu, Yiwu Yao, Sinno Jialin Pan, and Mingxuan Yuan. 2025. KVTuner: Sensitivity-aware layer-wise mixed-precision KV cache quantization for efficient and nearly lossless LLM inference. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024. Snapkv: Llm knows what you are looking for before generation.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. 2024. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*. PMLR.

OpenAI. 2024. API prompt caching. Accessed: 2024-12-25.

Ruoyu Qin, Zheming Li, Weiran He, Mingxing Zhang, Yongwei Wu, Weimin Zheng, and Xinran Xu. 2024. Mooncake: A kvcache-centric disaggregated architecture for llm serving.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Jintao Zhang, Jia Wei, Haofeng Huang, Pengle Zhang, Jun Zhu, and Jianfei Chen. 2025. Sageattention: Accurate 8-bit attention for plug-and-play inference acceleration. In *International Conference on Learning Representations (ICLR)*.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. 2023. H_2o: Heavy-hitter oracle for efficient generative inference of large language models.

Xingyu Zheng, Yuye Li, Haoran Chu, Yue Feng, Xudong Ma, Jie Luo, Jinyang Guo, Haotong Qin, Michele Magno, and Xianglong Liu. 2025. An empirical study of qwen3 quantization.