BLAZE: BLOCK-LEVEL ADAPTIVE KV QUANTIZATION GUIDED BY ATTENTION

Anonymous authors
Paper under double-blind review

ABSTRACT

The KV cache is the memory bottleneck of long-context inference and grows with sequence length. Attention concentrates on a few positions, yet compression methods treat every token alike or evict tokens irreversibly. We present Blaze, a block-level adaptive KV cache quantization scheme that derives the precision of each block from the attention its positions receive. During prefill, Blaze marks a position closed when its attention stays below a threshold that decays with layer depth, since deeper layers attend to fewer positions, and it requires agreement across consecutive layers, so single-layer fluctuations do not drive the decision. Blocks are stored in INT8 or INT4 according to how many of their positions closed, blocks that carry long-distance positions keep the higher precision, and every decision is fixed during prefill, so decoding reads a lookup table and adds $O(1)$ work per step. Across five models, Blaze compresses the KV cache by 70–71% at 4K and 8K prefixes at a perplexity within 1.6% of FP16, and it stays within 2 points of FP16 on eight downstream tasks and hard four-needle retrieval, where uniform INT4 falls 6 to 14 points behind. Blaze lowers the memory barrier for long-context inference by assigning precision to blocks instead of discarding tokens.

1 INTRODUCTION

The KV cache is the dominant memory bottleneck in autoregressive inference as sequence length increases. Unlike model weights, the KV cache scales linearly with sequence length, layer count, hidden dimension and batch size. For a 7B-parameter model with 32 layers and a 32K context, the KV cache takes approximately 16 GB (Hooper et al., 2024), a figure that assumes multi-head attention, while grouped-query attention makes the same cache three to four times smaller. It exceeds the memory footprint of model weights and forces reduced batch sizes, truncated contexts or model parallelism (Dao et al., 2022; Kwon et al., 2023). At 128K tokens, the same cache grows to 64 GB (Zhang et al., 2023).

Existing approaches fall into two categories: uniform quantization and token eviction. Methods like KVQuant (Hooper et al., 2024) and KIVI (Liu et al., 2024) quantize every token to the same bit-width, without considering how much attention each token receives. However, attention weights vary across positions: certain tokens receive high attention while others are largely ignored by most layers (Zhang et al., 2023; Xiao et al., 2024), so aggressive uniform quantization (e.g., INT4) degrades generation quality. Eviction-based methods, including H2O (Zhang et al., 2023), SnapKV (Li et al., 2024), and StreamingLLM (Xiao et al., 2024), reduce cache size by dropping tokens considered less important. This removal is irreversible: once evicted, a token's key and value states cannot be recovered, which degrades long-range dependencies and fine-grained retrieval over the full context (Li et al., 2024).

We introduce Blaze, a block-level adaptive KV cache quantization framework that assigns precision from the attention each block receives (Figure 1). Blaze avoids per-token decisions by operating on fixed blocks, and it protects long-distance information by excluding the most attended positions from closure voting. For the remaining blocks it applies cross-layer attention voting with a layer-dependent decaying threshold $\theta_\ell = \theta_{\text{base}} \times \alpha^\ell$: a block becomes INT4 when its tokens fall below the threshold across $K{=}3$ consecutive layers, and remains INT8 otherwise. The threshold decays because deeper layers attend to fewer tokens, matching their sparser attention patterns (Clark et al., 2019b). All decisions are made during prefill and stored in a lookup table, so decoding appends each new token at the precision of its block with $O(1)$ amortized computation per step.

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.

Our contributions are:

- **Problem analysis.** Uniform quantization applies one precision to every token, so saving memory on the tokens that keep receiving attention costs accuracy on those tokens. Eviction removes them from the cache instead. We measure how the attention received by a position varies across layers and use the agreement between consecutive layers as the signal for a block-level decision.
- **Method.** Blaze assigns one precision to each 64-token block. A position is marked closed when its attention stays below a layer-dependent decaying threshold in $K=3$ consecutive layers, a block with more than 80% closed positions is stored in INT4, and a block that carries a long-distance token beyond position 8,192 stays in INT8. All precisions are assigned during prefill and read from a lookup table during decoding, so each decode step adds $O(1)$ work.
- **Results.** Across five models, Blaze compresses the KV cache by 70–71% at 4K and 8K prefixes, 62% at 16K, and 56–57% at 32K, at a perplexity within 1.6% of FP16 on the 4B/7B/8B models. On eight downstream tasks and hard four-needle retrieval from 4K to 32K, accuracy stays within 2 points of FP16, where uniform INT4 loses 6 to 14 points at 16K.

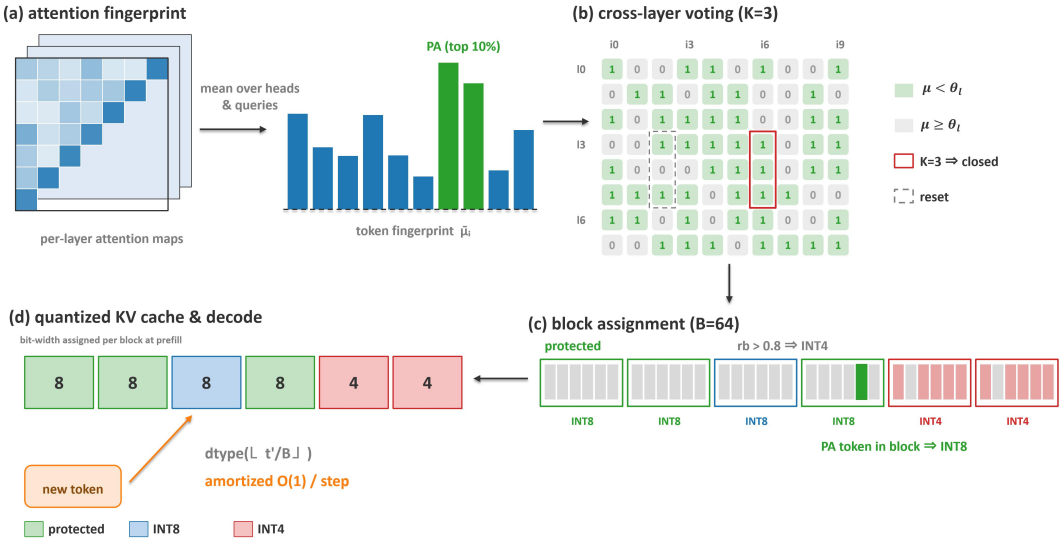

Figure 1: Overview of the Blaze framework. (a) Attention fingerprint extraction: per-layer attention maps are averaged into a per-position fingerprint. (b) Cross-layer voting with $K=3$ consecutive layers. (c) Block-level precision assignment: blocks are assigned INT8 or INT4. (d) Decode-stage update: each new token is appended at the precision of its block.

2 Related Work

Quantization approaches. KVQuant (Hooper et al., 2024), KIVI (Liu et al., 2024), and Atom (Zhao et al., 2024) quantize the KV cache (nearly) uniformly across tokens. KVQuant applies per-channel quantization to keys and per-token quantization to values. KIVI uses group-wise asymmetric quantization. Both assign the same precision to every token.

Mixed-precision allocation. A related line varies precision across the cache instead of applying one bit-width everywhere. ZipCache scores tokens by normalized attention and compresses the remaining ones harder (He et al., 2024), MiKV keeps important tokens at higher precision rather than evicting them (Yang et al., 2024), and SpectrumKV assigns a per-token precision level for prefill–decode transfer (Yang, 2026). RateQuant allocates bits across heads with a rate-distortion model (Zuo et al., 2026), and WindowQuant searches bit-widths per token window (Tao et al., 2026). Lee et al. (2026)

report that model-internal importance indicators agree with each other and that the benefit of mixing lies in interpolating the bit grid.

Eviction-based methods. H2O (Zhang et al., 2023) accumulates attention scores to identify "heavy hitter" tokens and evicts low-scoring entries. SnapKV (Li et al., 2024) prunes tokens with low attention scores in each layer. StreamingLLM (Xiao et al., 2024) keeps attention-sink tokens and evicts middle ones. These methods reduce memory, but the loss is irreversible.

Positioning of Blaze. Blaze classifies 64-token blocks instead of individual tokens, so its decisions map to hardware memory units, and it aggregates attention across layers, which avoids the noise of single-layer statistics. It compresses through adaptive precision rather than eviction, keeping every position in the cache at reduced precision, and it derives precision from attention statistics rather than from a fixed partition of positions. Prior mixed-precision methods score importance per token, per head, or per window, and several search over configurations. Blaze decides at block granularity from a vote over consecutive layers, with a fixed threshold and no calibration pass.

3 METHODOLOGY

3.1 OVERVIEW

Transformer language models (Vaswani et al., 2017) store key and value states of previous tokens in a KV cache that grows linearly with sequence length, becoming the primary memory bottleneck for long-context inference (Hooper et al., 2024; Wang et al., 2024). Existing quantization methods apply the same precision uniformly to every position, ignoring that some tokens attract sustained attention across layers while others are rarely queried after their initial appearance. Blaze instead assigns lower precision to positions whose attention contributions have fallen below a threshold, while protecting tokens that remain important.

Blaze assigns block-level precision in two phases. During prefill, we extract per-position attention scores from the full attention maps and mark positions as closed when their scores fall below layer-adaptive thresholds across $K = 3$ consecutive layers. We then partition the sequence into fixed 64-token blocks. We use a protection window $H = 128$: the first H tokens and the last H tokens are always kept in INT8. Blocks holding a protected long-distance token (index $\geq L_{\text{prot}}$) are kept in INT8 as well. For each remaining block, if more than 80% of its positions are closed, the entire block is assigned INT4. Otherwise it remains in INT8. During decode, each new token is appended to the cache at the precision of its block, which is INT8 beyond the prefix.

3.2 BLAZE CLOSURE DETECTOR

The Blaze Closure Detector identifies positions in the prefix whose attention contributions have fallen below a threshold across multiple layers. Closed positions receive INT4 quantization because their influence on future tokens is small, and the positions that stay open keep their block in INT8.

Attention fingerprint extraction. During prefill, the detector consumes the attention weights $\mathbf{A}^{(l)} \in \mathbb{R}^{h \times t \times t}$ of every layer $l \in \{0, \ldots, L-1\}$, where h is the number of attention heads and t is the sequence length. The fingerprint uses the column means of each layer, so the reduction runs over tiles as the attention weights are produced and no $h \times t \times t$ matrix is materialized. For each key position i and layer l, we compute the mean attention weight averaged across all heads h' and all query positions j:

$$\mu_i^{(l)} = \frac{1}{h \cdot t} \sum_{h'=0}^{h-1} \sum_{j=0}^{t-1} A_{h',j,i}^{(l)}. \tag{1}$$

The collection $\{\mu_i^{(l)}\}_{l=0}^{L-1}$ forms an attention fingerprint for position i. Under causal masking, the entries with $j < i$ are zero, so the average over all t queries is a fraction $(t-i)/t$ of the average over the queries that can attend to position i. The factor depends on position and not on layer, so it enters the PA ranking and the closure test in the same way.

Layer-wise decaying thresholds. Clark et al. (2019b) report that earlier layers produce broad, uniform attention patterns while deeper layers concentrate on sparse, semantically important tokens.

We define a per-layer closure threshold that decays exponentially with layer depth, $\theta_l = \theta_{\text{base}} \cdot \alpha^l$, with $\theta_{\text{base}} = 0.0004$ and $\alpha = 0.95$: deeper layers use lower thresholds, as their attention scores are sparser.

Long-distance token protection. We select the top 10% of positions by average attention across all layers as protected-attention (PA) tokens. When the sequence length exceeds $L_{\text{prot}} = 8192$, a block holding a PA token at position ≥ 8192 is kept in INT8, regardless of its closure ratio. PA tokens below 8192 do not force their block to INT8, and their block precision is set by the closure ratio. Since excluded positions are counted as open, a block holding PA tokens below 8192 needs a larger closed share to pass the 0.8 threshold.

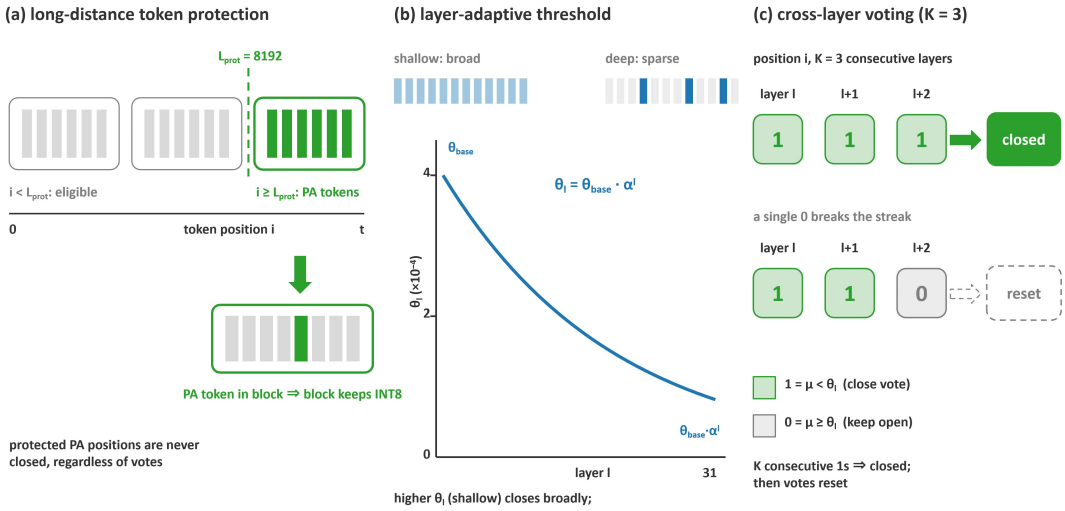

Figure 2: The Blaze Closure Detector. (a) Long-distance token protection: selected positions with index $\geq L_{\text{prot}}$ keep their block in INT8. (b) Layer-adaptive threshold θ_l. (c) Cross-layer voting with K=3 consecutive layers.

Cross-layer consistency voting. A position is marked closed when its attention score falls below the threshold across $K = 3$ consecutive layers (Figure 2c). We evaluate positions $i < t - H$ that are not selected for protection. For each evaluated position and layer l, we define $v_i^{(l)} = \mathbf{1}[\mu_i^{(l)} < \theta_l]$. If K consecutive layers satisfy $v_i^{(l')} = 1$, position i is marked closed. This prevents closure due to a temporary drop in one or two layers. Algorithm 1 in the appendix summarizes the procedure.

3.3 Block-Level Quantization Assignment

Processing each token separately would create too much metadata overhead and reduce memory access efficiency. Instead, we divide the sequence into blocks of $B = 64$ tokens, each receiving a single precision assignment. Given the closed set $\mathcal{C}$, we partition the sequence of length t into $N_b = \lceil t/B \rceil$ blocks. For block b spanning positions $[s_b, e_b)$, we compute the closure ratio $r_b = |\{i \in [s_b, e_b) : i \in \mathcal{C}\}|/(e_b - s_b)$.

Block assignment. Let r_b be the closure ratio of block b. We set precision(b) = INT4 when $r_b > 0.8$ and $b \notin \mathcal{B}_{\text{prot}}$, and INT8 otherwise. The 0.8 threshold ensures that a block is quantized to INT4 when most of its positions are closed, avoiding mixed-precision artifacts within a block. Blocks that contain the first H tokens, the last H tokens, or a PA token at position ≥ 8192 are always kept in INT8.

Compression ratio. Let N_{int8} and N_{int4} denote the number of tokens stored in INT8 and INT4, respectively. With $b_{\text{fp16}} = 2$, $b_{\text{int8}} = 1$, and $b_{\text{int4}} = 0.5$ bytes per element, the effective compression

ratio relative to FP16 is:

$$\eta = 1 - \frac{N_{\text{int8}} \cdot b_{\text{int8}} + N_{\text{int4}} \cdot b_{\text{int4}}}{t \cdot b_{\text{fp16}}}. \tag{2}$$

The hidden dimension d appears in every term and cancels out. The accounting covers the prefilled tokens and excludes the quantization metadata, such as per-group scales and zero points.

3.4 Decode-Stage Update and Complexity

Blaze performs $O(1)$ work per step during decode, where re-quantizing the entire cache at every step would require $O(t)$ operations per step. The new token occupies position t' and its block index is $b^* = \lfloor t'/B \rfloor$. Blocks inside the prefix carry a precision fixed during prefill, and blocks beyond the prefix default to INT8, since new positions fall in the protection window. Each step appends the new key and value states and touches one block, so the per-step cost is $O(1)$. In the prototype, positions appended during decoding are held in FP16 while they fall inside the newest H positions and are quantized to INT8 once they leave that window. During prefill, the detector reads the L attention tensors from the standard forward pass in tiles, accumulating the column means of each layer. Fingerprint aggregation scans every entry of the L attention maps and costs $O(LHt^2)$ operations. The vote adds $O(Lt)$ operations. Standard prefill attention costs $O(Lt^2d)$ operations, which dominates the fingerprint work, so the extra cost does not change the prefill complexity.

3.5 Experimental Setup

We evaluate Blaze on five open-weight language models: Qwen3-4B-Instruct, Qwen3-8B, Qwen3-30B-A3B-Instruct, Llama-3.1-8B-Instruct (Qwen Team, 2025; Grattafiori et al., 2024), and Mistral-7B-Instruct-v0.3 (Jiang et al., 2023). Perplexity (PPL) is measured on WikiText-2 (Merity et al., 2017) with prefix lengths from 2,048 to 32,768 tokens and a generation length of 256.

For downstream evaluation, we select six zero-shot commonsense tasks from lm-evaluation-harness (Gao et al., 2023): ARC-Easy, ARC-Challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), WinoGrande (Sakaguchi et al., 2020), and BoolQ (Clark et al., 2019a), each with 300 samples. Beyond commonsense, we include MMLU (STEM subset, 300 questions, 5-shot) (Hendrycks et al., 2021) and GSM8K (300 questions, 8-shot) (Cobbe et al., 2021).

Long-context retrieval uses two evaluations on Qwen3-4B-Instruct and Llama-3.1-8B-Instruct. The first evaluation extends the RULER multi-value task by placing four needles in the same key paragraph. Haystacks come from WikiText-2 at lengths from 4,096 to 32,768 tokens. A sample is correct when all four needles are recovered, and every needle is also scored on its own. The second evaluation is RULER itself (Hsieh et al., 2024): multi-key retrieval, multi-value retrieval, and variable tracking at 4,096–16,384 tokens under the official string-match-all metric.

Baselines. FP16 is the baseline, alongside uniform INT8 (per-channel) and INT4 (group size 32) quantization. For prior quantization methods we evaluate KIVI (Liu et al., 2024) and KVQuant (Hooper et al., 2024), both re-implemented in the same setup. KIVI runs in its INT4 group-wise configuration with group size 32. Under Eq. 2, the compression of an all-INT4 cache is 75.0%. Its 2-bit configuration would give 87.5% and is not simulated here. KVQuant runs in the per-channel INT8 configuration of its paper, which yields 50.0% compression. Blaze's compression is computed from its per-block assignment.

Implementation. Experiments use PyTorch (Paszke et al., 2019; Jacob et al., 2018) on NVIDIA A100-SXM4-80GB GPU. We use per-channel INT8 and group-wise INT4 (group size 32) quantization, implemented as fake quantization, and compared with an integer-rounded path in Section 3.11. Hyperparameters are: $\theta_{\text{base}} = 0.0004$, $H = 128$, $B = 64$, $K = 3$, $\alpha = 0.95$, $\rho = 0.10$, and $L_{\text{prot}} = 8192$.

3.6 Downstream Task Accuracy

Table 1 reports zero-shot accuracy on eight downstream tasks for the four models and the four cache settings.

Blaze stays within 1.7 points of FP16 on any single task and within 0.2 points in the eight-task average, while the worst single-task drop of KIVI, the strongest compressed baseline, is 3.0 points.

Table 1: Zero-shot downstream accuracy (%) with a compressed KV cache, 300 samples per task: the six commonsense tasks, MMLU (STEM, 5-shot), and GSM8K (8-shot). KIVI (INT4, group size 32) and KVQuant (per-channel INT8) quantize the prefilled cache and keep every generated token in FP16. Compression follows Eq. 2: 70.2–70.5% for Blaze at a 4K prefix, 75.0% for KIVI, and 50.0% for KVQuant. Blaze values at or above FP16 are bold.

	Qwen3-4B				**Qwen3-8B**				**Llama-3.1-8B**				**Mistral-7B**			
Task	FP16	**Blaze**	KIVI	KVQuant	FP16	**Blaze**	KIVI	KVQuant	FP16	**Blaze**	KIVI	KVQuant	FP16	**Blaze**	KIVI	KVQuant
ARC-Easy	69.7	68.7	66.7	69.0	78.0	**78.0**	78.0	78.0	73.7	**74.7**	74.7	73.7	86.0	**86.3**	85.7	86.3
ARC-Challenge	62.7	61.0	63.3	62.7	72.7	**73.7**	73.3	73.3	66.3	65.0	67.7	66.7	72.0	**72.3**	72.0	72.0
HellaSwag	33.0	32.7	33.3	32.3	34.3	**34.3**	34.3	34.3	31.3	**31.7**	30.3	31.0	45.7	45.3	45.7	45.7
PIQA	79.7	**79.7**	80.0	80.3	78.3	**78.7**	79.3	78.3	81.3	81.0	82.3	81.7	82.0	81.7	81.7	82.0
WinoGrande	59.3	**60.7**	58.7	60.7	60.7	59.7	59.7	59.7	60.7	59.7	59.3	60.7	66.0	**66.0**	65.3	65.3
BoolQ	85.7	**86.0**	85.3	86.7	86.3	**86.7**	86.7	87.0	79.7	**79.7**	78.7	79.7	87.0	**87.3**	87.7	87.0
Average (commonsense)	65.0	64.8	64.6	65.3	68.4	**68.5**	68.6	68.4	65.5	65.3	65.5	65.6	73.1	**73.2**	73.0	73.1
MMLU (STEM)	72.7	71.7	72.0	72.3	71.7	70.3	69.7	71.3	58.3	59.0	57.3	58.3	53.7	53.0	53.7	53.7
GSM8K	82.0	82.7	84.7	82.7	92.0	91.3	91.7	91.3	77.3	77.3	78.0	77.7	52.3	51.3	50.7	52.0
Average (MMLU, GSM8K)	77.3	77.2	78.3	77.5	81.8	80.8	80.7	81.3	67.8	**68.2**	67.7	68.0	53.0	52.2	52.2	52.8

Blaze holds this accuracy across four models at a compression 20 points above the INT8 configuration of KVQuant, and within 5 points of KIVI.

3.7 Perplexity and Compression

Figure 9 in the appendix plots the compression–quality trade-off for FP16, Blaze, KIVI, KVQuant, and the uniform baselines on the five models at 4K prefixes, and Table 12(a) gives their paired statistics. Blaze compresses 70.2–70.5% with a paired per-sample ΔPPL within 1.5% of FP16 on the four 4B/7B/8B models and within 2.1% on Qwen3-30B, measured on the same samples as KIVI, KVQuant, and the uniform baselines.

3.8 Long-Context Evaluation

Figure 5 in the appendix plots compression and the paired ΔPPL for the five models at prefix lengths from 2,048 to 32,768 tokens. Compression is 70.2–71.0% from 4K to 8K and falls to 56–57% at 32K, because beyond 8K every block that holds a protected long-distance token stays in INT8. Across the twenty 4B/7B/8B runs the paired ΔPPL stays within 1.6% of FP16, and on Qwen3-30B within 2.6%.

3.9 Exact Retrieval: NIAH and RULER

Perplexity reflects average performance, but it does not show whether compression discards information needed for a distant retrieval. Figure 3 reports per-needle recall on the four-needle task, where a sample is correct when all four needles in the same key are recovered, and Table 13 in the appendix gives the all-needle accuracy behind those cells.

On both models and at every length, Blaze stays within 2 points of FP16, and per-needle retrieval stays above 97% with and without PA protection. Removing PA protection changes all-needle accuracy by at most three samples per cell, and the no-PA variant is ahead at 32K. Uniform INT4 loses 6 to 14 points at 16K at the same 75% compression, and needles inside INT4 blocks are retrieved at rates similar to needles in protected blocks. Figure 3 reports per-needle recall by depth: Blaze misses at most two needles in any bin on Llama-3.1-8B at 32K, while uniform INT4 loses needles across most depth bins at 16K and 32K.

We also evaluate RULER synthetic tasks (Hsieh et al., 2024): multi-key retrieval, multi-value retrieval, and variable tracking. Table 2 reports per-task averages across 4K–16K, and no Blaze score falls more than 1.0 point below FP16.

3.10 Ablation Study

Table 3 reports the ablation on Qwen3-4B at a 16K prefix. Blaze reaches the lowest PPL of the six configurations at 62.1% compression. Removing PA protection raises compression to 74.6% and extending protection to position 0 lowers it to 53.3%, while PPL changes by at most 0.05. The Fixed Zone baseline assigns INT8 to the first 256 tokens and then alternates INT4 and INT8 across 64-token

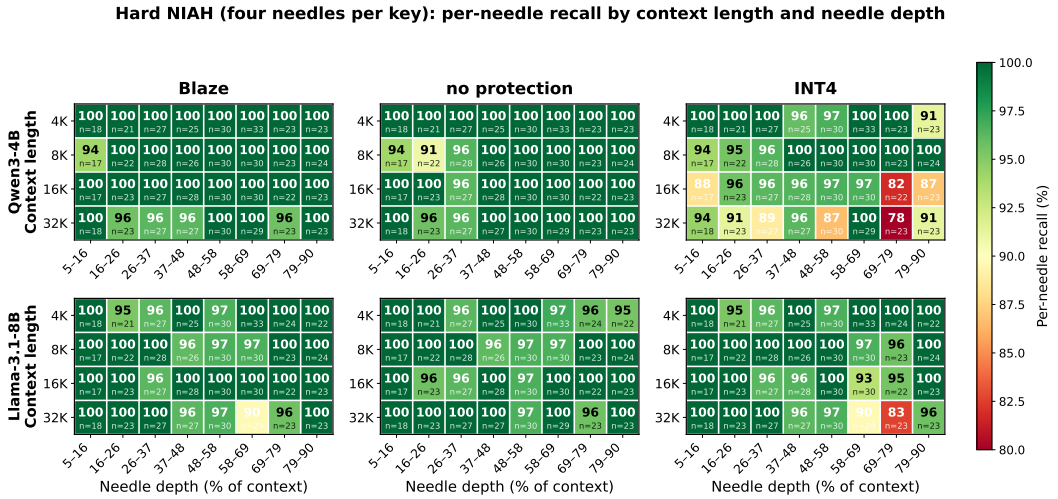

Figure 3: Hard NIAH per-needle recall (%) by context length (rows) and needle depth (columns). Needles are placed at 5–90% of the context. Results for Blaze, for Blaze without PA protection, and for uniform INT4 on Qwen3-4B (top) and Llama-3.1-8B (bottom). n is the number of needles in each bin (200 per length).

Table 2: RULER synthetic task scores (%). Per-task averages across 4K–16K. Single-key retrieval is omitted. Blaze values at or above FP16 are bold.

Model	Task	FP16	**Blaze**	KIVI
Qwen3-4B	NIAH multi-key	100.0	**100.0**	99.3
Qwen3-4B	NIAH multi-value	98.9	98.8	98.3
Qwen3-4B	Variable tracking	100.0	99.9	99.7
Llama-3.1-8B	NIAH multi-key	99.7	**100.0**	99.3
Llama-3.1-8B	NIAH multi-value	97.3	**98.1**	97.2
Llama-3.1-8B	Variable tracking	99.9	**99.9**	99.7

blocks by parity, and at Blaze's compression it is 0.26 PPL worse. The uniform baselines at 50% and 75% compression also have higher PPL than Blaze.

Restricting protection to positions above 8192 raises compression to Blaze's 62.1% at 16K, and the random-block control in Appendix D matches Blaze's accuracy at the same protected-block count.

Threshold sensitivity. We sweep θ_{base} on Qwen3-4B at prefix 2,048 (Figure 6 in the appendix). At $\theta_{\text{base}} \leq 0.0002$ the detector closes almost no blocks and Blaze collapses to the uniform-INT8 level of 50.0% compression, while at $\theta_{\text{base}} \geq 0.0008$ compression saturates at 69.3% with no further PPL improvement. The remaining tokens appear important and resist closure even under aggressive thresholds.

At 16K, $K \in \{1, 3, 5\}$ and $\alpha \in \{0.90, 0.99\}$ give the same average PPL and compression ratio, so the assignment is insensitive to K and α in this range. The protected region size H has little effect: across $H \in \{64, 128, 256\}$, PPL varies by at most 0.01 and compression by under 1.4 points (Table 8). The threshold sweep and the H sweep use 20 and 56 samples, so their FP16 baselines differ from the long-context runs, which use 50 samples per cell. The default $\theta_{\text{base}} = 0.0004$ is the smallest value at which the detector closes any block.

3.11 Robustness and Validation

Long-generation robustness. The PPL gap to FP16 does not widen as generation grows from 256 to 1,024 tokens, staying between 0.30% and 1.43% below FP16 (Table 5), and the per-step loss change

Table 3: Ablation study on Qwen3-4B (prefix 16,384, 256 generated tokens, 50 samples). The Fixed Zone is a fixed partition with no attention feedback.

Configuration	Avg PPL	Compress
Blaze (protection $\geq$ 8192)	**10.05**	**62.1%**
Blaze, PA protection from position 0	10.10	53.3%
Blaze without PA protection	10.07	74.6%
Fixed Zone	10.31	62.3%
Uniform INT8	10.16	50.0%
Uniform INT4	10.55	75.0%

from the first half to the second half of generation is no larger for Blaze than for FP16 (Table 9), so Blaze does not accumulate quantization error during decoding.

Cross-dataset generalization. On C4 and WikiText-2 the default threshold holds without retuning, at 71.0% compression on C4 with PPL within 0.8% of FP16 on both corpora (Table 6).

Downstream tasks. Across the four models and eight tasks of Table 1, no compressed method falls more than 3.0 points below FP16, and no eight-task average moves by more than 0.3 points.

Batch-size scaling. Compression stays at 70–71% and the paired ΔPPL changes by at most 1.7% as the batch size grows from 1 to 8 (Table 10).

3.12 Efficiency

Blaze's measured throughput is lower than FP16 (5.7 vs. 19.3 tokens/s at batch 1 on Qwen3-4B) because fake quantization adds overhead, and the compression ratios above are projected cache footprints under INT8/INT4 storage.

On the GPU side the detection pass costs between 0.8 and 1.4 plain prefill passes on Qwen3-4B (Figure 7a,b). The end-to-end prototype prefill is far slower than this accounting: 124 s at 16K against 7.1 s for a plain prefill, because the prototype assigns precisions with a Python loop over tokens rather than in a kernel. The decode phase adds an $O(1)$ table lookup per step.

On the memory side, the measured memory traffic per decode step is 50.8% of FP16 for INT8 and 28.1% for INT4, and under projected INT8/INT4 storage Blaze's footprint follows its measured compression at each length (Figure 7c). At 32K tokens on Qwen3-4B, Blaze requires 1.93 GiB against 4.50 GiB for FP16.

4 Discussion

Cross-layer voting. Transformer attention patterns are stable across consecutive layers. Adjacent-layer fingerprints μ_i correlate at 0.819, while correlation with layer 0 drops to 0.13–0.25 by the middle layers (Figure 8). This local consistency supports $K{=}3$ consecutive voting: it pools nearby layers and reduces transient low-attention spikes that could trigger false positives in single-layer detectors.

Protection cost at long prefixes. Because the selection covers a fixed 0.10 of all positions and protection applies beyond 8,192, the protected footprint grows with the long-distance part of the prefix. Table 4 gives the compression this costs: about 3 points up to 8K, 12.4 points at 16K, and 17.4 to 17.7 points at 32K. Protection earns its cost while it stays selective: it pays at 8K, while at 32K it is 4 points below the same run without protection. A budget that holds the protected share of blocks constant, rather than a fixed share of selected positions, would recover this cost, and the random-block control in Appendix D shows that the effect follows from how many blocks are held in INT8 rather than from which are selected.

Limitations. The experiments use fake quantization. Real deployment requires custom CUDA kernels that support native INT4/INT8 storage and fused dequantization during attention, similar to FlashAttention (Dao et al., 2022) and vLLM (Kwon et al., 2023). The detector also needs per-column

Table 4: Long-distance protection on the four-needle task, Qwen3-4B and Llama-3.1-8B, 50 samples per cell (compression ranges cover the two models). Protection holds selected positions beyond 8,192 in INT8, and the cost is the difference against the same run without it. Accuracy is all-needle exact match, shown as with → without protection. The other cache settings for these runs are in Table 13.

	Compression (%)			All-needle accuracy (%)	
Prefix length	With	Without	Cost (pts)	Qwen3-4B	Llama-3.1-8B
4,096	69.8–70.0	73.1	3.1–3.2	100 → 100	98 → 96
8,192	70.6–70.7	74.0	3.3–3.4	98 → 92	98 → 98
16,384	62.1	74.5	12.4	100 → 98	98 → 96
32,768	57.1–57.3	74.8	17.4–17.7	92 → 96	92 → 96

attention statistics, which standard FlashAttention kernels do not expose, so a deployment would add a statistics output to the kernel or run the fingerprint on a slower attention path. Hardware compatibility is another constraint: FP4 requires SM100 or later (Blackwell), while A100 supports FP8 E5M2. The attention fingerprint averages over all query positions, which scales μ_i by $(t - i)/t$ (Section 3.2).

Future directions. Real INT4/INT8 kernels integrated with vLLM and SGLang would unlock Blaze's latency benefits. An FP8 E5M2 variant could extend compatibility to A100-class GPUs. Dynamic threshold adaptation might improve robustness for ultra-long contexts. Extending Blaze to multi-modal KV caches is another direction.

5 Conclusion

We present Blaze, a block-level adaptive KV cache quantization framework that assigns per-block precision through cross-layer attention voting. Across five language models, Blaze compresses KV caches by 70–71% at 4K with a paired perplexity difference within 1.5% of FP16 on four models and 2.1% on Qwen3-30B. On eight downstream tasks and four models, accuracy changes by at most 0.2 points on average and 1.7 points on a single task, where the largest drop of KIVI is 3.0 points and the INT8 configuration of KVQuant compresses 20 points less.

On sequences up to 32,768 tokens, compression falls from 70–71% at 4K and 8K to 62% at 16K and 56–57% at 32K, with the paired perplexity difference within 1.6% of FP16 on the 4B/7B/8B models. Blaze stays within 2 points of FP16 on four-needle retrieval from 4K to 32K, where uniform INT4 loses 6 to 14 points at 16K, and no RULER score is more than 1.0 point below FP16. Fake and integer-rounded quantization give identical values, and each decode step adds $O(1)$ amortized overhead.

Blaze reduces KV cache memory with little change in model quality, lowering the memory barrier for longer-context inference on memory-limited hardware.

AI use statement

According to the ICLR 2027 policy on AI assistance, we disclose our use of Large Language Models (LLMs) as follows. We used LLMs to aid and polish the writing of the paper, for language-level editing of author-written drafts such as grammar correction, phrasing, and conciseness. We used LLMs for retrieval and discovery of related work, for example by suggesting candidate papers from keyword queries. LLMs were not used for the remaining tasks with required disclosure, and we did not use LLMs to generate experimental results, code, or figures. Every reference cited in this paper was retrieved, read, and verified by the authors, and the selection and discussion of related work are the authors' own. All technical content, claims, methodology, and results originate from the authors' own work and were verified by the authors. We have reviewed all AI-assisted work and take responsibility for the final content of this work, including text, claims, and artifacts.

REFERENCES

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. PIQA: Reasoning about physical commonsense in natural language. In *AAAI Conference on Artificial Intelligence*, 2020.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2019a.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 2019b.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

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. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation. `https://zenodo.org/records/10256836`, 2023. Zenodo.

Aaron Grattafiori et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

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 (NeurIPS)*, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations (ICLR)*, 2021.

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

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

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *ACM Symposium on Operating Systems Principles (SOSP)*, 2023.

Daeha Lee, Do-Hyung Kim, and Jae-Hong Kim. SemKV: Semantic mixed-precision KV cache quantization guided by the quality cliff for long-context LLM inference. *arXiv preprint arXiv:2608.28911*, 2026.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

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

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

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

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

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. In *AAAI Conference on Artificial Intelligence*, 2020.

Wei Tao, Xiaoyang Qu, Peiqiang Wang, Guokuan Li, Jiguang Wan, Kai Lu, et al. WindowQuant: Mixed-precision KV cache quantization based on window-level similarity for VLMs inference optimization. *ACM Transactions on Architecture and Code Optimization (TACO)*, 2026.

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 (NeurIPS)*, 2017.

Zihao Wang, Bin Cui, and Shaoduo Gan. Squeezeattention: 2d management of KV cache in LLM inference via layer-wise optimal budget. *arXiv preprint arXiv:2404.04793*, 2024.

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

June Yong Yang, Byeongwook Kim, Jeongin Bae, Beomseok Kwon, Gunho Park, Eunho Yang, et al. No token left behind: Reliable KV cache compression via importance-aware mixed precision quantization. *arXiv preprint arXiv:2402.18096*, 2024.

Pengju Yang. SpectrumKV: Per-token mixed-precision KV cache transfer for prefill-decode disaggregated LLM serving. *arXiv preprint arXiv:2606.08635*, 2026.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Yilong Zhao, Chien-Yu Lin, Kan Zhu, Zihao Ye, Lequn Chen, Size Zheng, Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, and Baris Kasikci. Atom: Low-bit quantization for efficient and accurate LLM serving. In *Machine Learning and Systems (MLSys)*, 2024.

Fei Zuo, Zikang Zhou, Hao Cong, Xiaoyan Xi, and Ho Fai Leung. RateQuant: Optimal mixed-precision KV cache quantization via rate-distortion theory. *arXiv preprint arXiv:2605.06675*, 2026.

A Additional Method Details

Figure 4 illustrates the block-level precision assignment and decode-stage cache update. In (a), a block with closure ratio $r_b > 0.8$ is assigned INT4, otherwise INT8. Boundary tokens and blocks holding a protected PA token at position $\geq L_{\text{prot}}$ are kept in INT8. In (b), each new token is appended at the precision of its block. Algorithm 1 gives the detection procedure and Algorithm 2 the cache update.

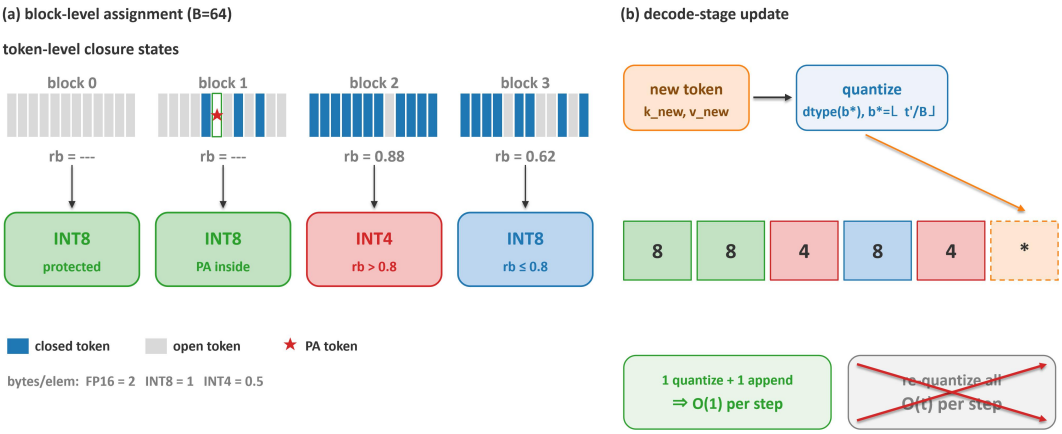

Figure 4: (a) Block-level precision assignment from the closure ratio, with boundary tokens and protected PA blocks kept in INT8. (b) Decode-stage update, where the new token is appended at the precision of its block.

A.1 Long-Context Trends and Long-Generation Robustness

Figure 5 plots the long-context trends. Table 5 extends generation from 256 to 512 and 1,024 tokens and measures PPL on the generated tokens. Blaze improves PPL across all four settings by between 0.30% and 1.43%. On Qwen3-8B with 512-token generation, the per-token loss change from the first half to the second half is -0.123 for FP16 and -0.115 for Blaze.

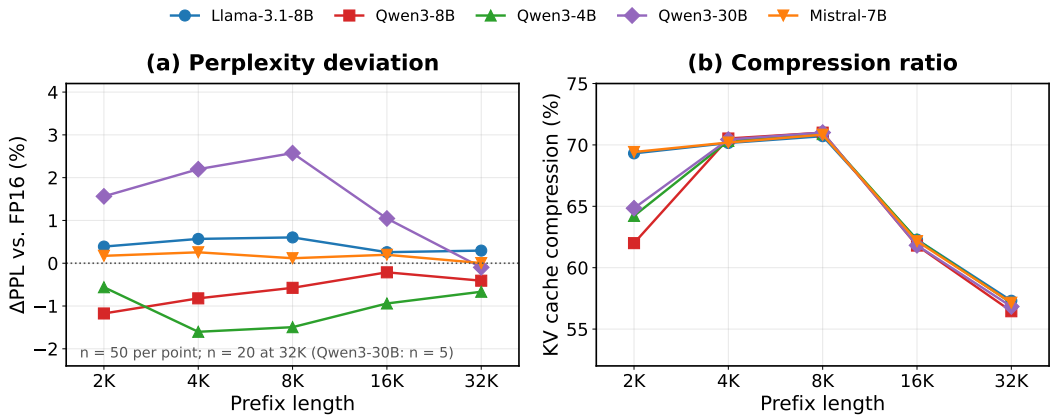

Figure 5: Long-context behavior. (a) ΔPPL relative to FP16 across prefix lengths. (b) Compression ratio across prefix lengths.

Algorithm 1 Blaze Closure Detection

Require: Attention maps $\{\mathbf{A}^{(l)}\}_{l=0}^{L-1}$, sequence length t, base threshold θ_{base}, decay α, protection window H, consistency K, protection ratio ρ, long-distance threshold L_{prot}, block size B
Ensure: Closed positions $\mathcal{C} \subseteq \{0, \ldots, t-1\}$, protected blocks $\mathcal{B}_{\text{prot}}$

$\theta_l \leftarrow \theta_{\text{base}} \cdot \alpha^l$ for $l = 0, \ldots, L-1$
for each layer l and each position $i \in \{0, \ldots, t-1\}$ **do**
  $\mu_i^{(l)} \leftarrow \text{mean}_{h',j}\ A_{h',j,i}^{(l)}$
end for
$\bar{\mu}_i \leftarrow \frac{1}{L}\sum_l \mu_i^{(l)}$
$\mathcal{S} \leftarrow$ top-ρ positions by $\bar{\mu}_i$ {PA token set, selected globally}
if $t > L_{\text{prot}}$ **then**
  $\mathcal{P} \leftarrow \{i \in \mathcal{S} : i \geq L_{\text{prot}}\}$ {long-distance protection set}
  $\mathcal{B}_{\text{prot}} \leftarrow \{b : [s_b, e_b) \cap \mathcal{P} \neq \emptyset\}$ {blocks held in INT8}
else
  $\mathcal{B}_{\text{prot}} \leftarrow \emptyset$
end if
for each position $H \leq i < t - H$ and each layer l **do**
  $v_i^{(l)} \leftarrow \mathbf{1}[\mu_i^{(l)} < \theta_l]$
end for
$\mathcal{C} \leftarrow \emptyset$
for each position $H \leq i < t - H$ with $i \notin \mathcal{S}$ **do**
  $c \leftarrow 0$
  for layers $l = 0, \ldots, L-1$ in order **do**
    if $v_i^{(l)} = 1$ **then**
      $c \leftarrow c + 1$
      if $c \geq K$ **then**
        $\mathcal{C} \leftarrow \mathcal{C} \cup \{i\}$
        break
      end if
    else
      $c \leftarrow 0$
    end if
  end for
end for
return $\mathcal{C}$, $\mathcal{B}_{\text{prot}}$

Algorithm 2 BlazeCache Update

Require: New key $\mathbf{k}_{\text{new}}$, value $\mathbf{v}_{\text{new}}$, layer index l, block precisions $\{\text{precision}[b]\}_{b=0}^{N_b-1}$, block size B
Ensure: Updated quantized cache $(\mathbf{K}_q^{(l)}, \mathbf{V}_q^{(l)})$

$t' \leftarrow$ current sequence length
$b^* \leftarrow \lfloor t'/B \rfloor$ {block index of the new token}
$d^* \leftarrow \text{precision}[b^*]$ if $b^* < N_b$, else INT8 {blocks beyond the prefix stay in INT8}
$\mathbf{k}_{\text{quant}} \leftarrow \text{quantize}(\mathbf{k}_{\text{new}}, d^*)$
$\mathbf{v}_{\text{quant}} \leftarrow \text{quantize}(\mathbf{v}_{\text{new}}, d^*)$
$\mathbf{K}_q^{(l)} \leftarrow \text{concat}(\mathbf{K}_q^{(l)}, \mathbf{k}_{\text{quant}})$
$\mathbf{V}_q^{(l)} \leftarrow \text{concat}(\mathbf{V}_q^{(l)}, \mathbf{v}_{\text{quant}})$
return $(\mathbf{K}_q^{(l)}, \mathbf{V}_q^{(l)})$

A.2 Cross-Dataset Generalization

Table 6 reports Blaze results on WikiText-2 and C4 with the same hyperparameters. On WikiText-2, Blaze reaches PPL 7.40 at 60.7% compression, compared with 7.41 for FP16. On C4, Blaze reaches PPL 11.73 at 71.0% compression, compared with 11.63 for FP16.

Table 5: Long-generation evaluation. Prefixes are 3,584 tokens for the 512-token runs, and 1,024 for Qwen3-4B and 3,072 for Qwen3-8B at 1,024 tokens (15 samples per row, 3 for Qwen3-4B at 1,024). PPL is measured on the generated tokens.

Model	Gen Len	FP16	Blaze	ΔPPL
Qwen3-4B	512	9.16	9.03	-1.43%
Qwen3-4B	1024	8.16	8.09	-0.87%
Qwen3-8B	512	8.70	8.66	-0.48%
Qwen3-8B	1024	8.18	8.15	-0.30%

Table 6: Cross-dataset evaluation on Qwen3-4B (generation length 256, 5 samples on C4 and 3 on WikiText-2, with prefixes of 3,840 and 1,792 tokens).

Dataset	FP16	Blaze	ΔPPL	Compress
WikiText-2	7.41	7.40	-0.04%	60.7%
C4	11.63	11.73	$+0.78\%$	71.0%

A.3 Quantization Rounding Validation

Table 7 compares fake quantization (quantize-then-dequantize in FP16) with an integer-rounded path (values cast to INT8/INT4 integers with native PyTorch integer types, then dequantized for compute) on three models. The two paths produce identical PPL values (to 10^{-4}) and identical compression ratios on all three models. The integer-rounded path adds an explicit cast to an integer type before dequantization, so the agreement is a check on the cast in the quantizer rather than a separate estimate of quantization error.

Table 7: Fake quantization vs. integer-rounded path (generation length 256, 5 samples per model, with prefixes of 3,840 tokens for the Qwen3 models and 1,792 for Llama-3.1-8B). Baseline PPL is measured within each run.

Model	Method	PPL	Compress
Qwen3-4B	Baseline FP16	8.47	—
	Fake Quant	8.39	70.7%
	Integer-rounded	8.39	70.7%
Qwen3-8B	Baseline FP16	7.46	—
	Fake Quant	7.47	71.0%
	Integer-rounded	7.47	71.0%
Llama-3.1-8B	Baseline FP16	6.12	—
	Fake Quant	6.12	69.4%
	Integer-rounded	6.12	69.4%

A.4 Threshold and Protected-Region Sensitivity

Figure 6 reports the base threshold sweep on Qwen3-4B (prefix 2,048, generation length 256). Table 8 varies the protection window H on Qwen3-4B (prefix 4,096). $H = 128$ achieves the lowest PPL with 70.3% compression. Compression follows Eq. 2 from the share of blocks in INT4: 18.1 of 32 blocks gives 64.1%, and 51.9 of 64 blocks gives 70.3%.

A.5 Decode Stability

Table 9 compares per-step cross-entropy loss between FP16 and Blaze on Qwen3-4B (prefix 4,096, generation length 256). The change from the first half to the second half is 0.0074 for Blaze and 0.0105 for FP16.

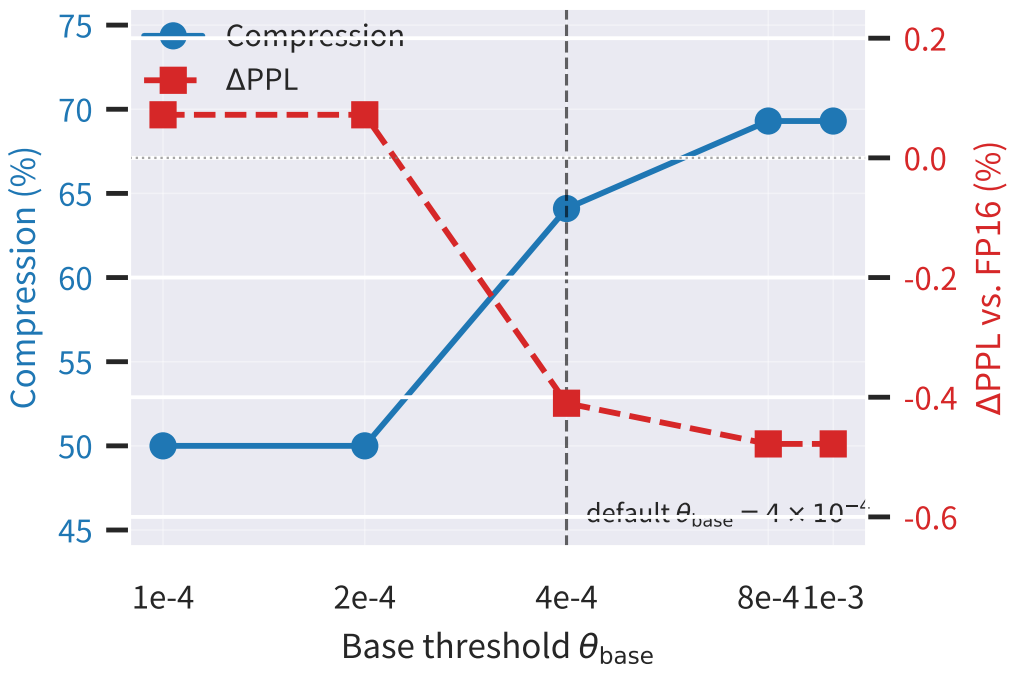

Figure 6: Compression ratio and ΔPPL relative to FP16 versus the base threshold θ_{base} (Qwen3-4B, prefix 2,048). The dashed line marks the default $\theta_{\text{base}} = 4 \times 10^{-4}$.

Table 8: Effect of protected region size H (Qwen3-4B, prefix 4,096, 56 samples). INT4 blocks is the mean number of the 64 prefix blocks assigned INT4.

H	PPL	Compress	INT4 blocks
64	10.24	70.7%	52.9/64
128	**10.23**	**70.3%**	**51.9/64**
256	10.24	69.3%	49.5/64

A.6 Batch-Size Scaling

Table 10 reports Blaze at batch sizes 1, 4, and 8 with prefix 4,096 and generation length 256. On Qwen3-4B, Blaze stays within 1.63% of FP16 PPL and compresses 70.2–70.4% at the three batch sizes. On Qwen3-8B, Blaze stays within 0.47% of FP16 PPL and compresses 70.6–71.0%.

B Efficiency Measurements

Figure 7 profiles the one-time detection pass on Qwen3-4B (a,b) and the projected KV cache footprint under INT8/INT4 storage (c). At 4K, Blaze trails uniform INT4 and KIVI by 4.7 points of compression. At 32K, Blaze requires 1.93 GiB, compared with 4.50 GiB for FP16 and 2.25 GiB for uniform INT8.

C Cross-Layer Stability Analysis

On Qwen3-4B at prefix 4,096, we compute the average received attention μ_i for each position at every layer and correlate the per-layer μ sequences on 20 WikiText-2 samples. Figure 8 plots the results. The Spearman correlation between adjacent layers $(\ell, \ell + 1)$ averages 0.819 across 700 layer pairs. The correlation between layer 0 and layer ℓ drops to 0.13–0.25 by the middle layers. The high

Table 9: Decode-stage stability on Qwen3-4B (prefix 4,096, generation length 256).

Metric	Baseline FP16	Blaze
First-half mean loss	2.2796	2.2713
Second-half mean loss	2.2900	2.2787
Drift (2nd – 1st)	0.0105	**0.0074**
Trend slope	0.000076	0.000070
Max loss	3.2368	3.2577
Min loss	1.4348	1.3999

Table 10: Batch-size scaling on Qwen3-4B and Qwen3-8B

Model	Batch	FP16	Blaze	ΔPPL	Compress
Qwen3-4B	1	9.02	8.91	-1.22%	70.4%
Qwen3-4B	4	8.66	8.52	-1.63%	70.2%
Qwen3-4B	8	8.45	8.32	-1.52%	70.4%
Qwen3-8B	1	8.24	8.22	-0.30%	70.6%
Qwen3-8B	4	7.94	7.90	-0.47%	70.7%
Qwen3-8B	8	7.75	7.75	-0.02%	71.0%

local correlation supports the $K{=}3$ consecutive-layer voting window, and the drop across depth is consistent with the layer-wise decaying threshold.

D Random-Block Control for Long-Distance Protection

Table 11 repeats three cells of the four-needle task with a control that holds the same number of blocks in INT8 as Blaze but chooses them at random. The control keeps the protected block count per sample equal to Blaze, so the two configurations have the same compression and differ in which blocks the protection covers. Both return the same all-needle accuracy and the same per-needle recall on all 150 samples, with no discordant sample. On these cells, a random partition of the same size is not distinguishable from the attention-based selection of protected blocks.

Table 11: Random-block control (four needles per key, $n{=}50$ per cell, 64 generated tokens, $\theta_{\text{base}} = 0.0004$, same protocol as Table 13). The control selects the same number of protected blocks per sample at random positions. Compression is computed from the per-block precision assignment and is identical in the two configurations, at 70.6%, 62.1%, and 62.1% for the three cells. Per-needle recall is 99.5%, 100.0%, and 99.5% in both.

Model	Length	Blaze (PA)	Random blocks	Compress
Qwen3-4B	8192	98.0	98.0	70.6%
Qwen3-4B	16384	100.0	100.0	62.1%
Llama-3.1-8B	16384	98.0	98.0	62.1%

E Statistical Summary

Table 12 collects the error of each measurement that the main text compares. PPL varies by 3.8–4.5 points across samples on the ablation and sweep runs, which is far wider than the differences between settings, so perplexity is compared through paired per-sample differences (spread 0.1–4.9%) and we report the standard error of their mean. Accuracy is compared against the binomial standard error of the benchmark.

Every Blaze–FP16 accuracy difference in Table 1 falls within one binomial standard error of the benchmark. In panel (a), Blaze is the setting closest to FP16 in mean paired ΔPPL on all five models,

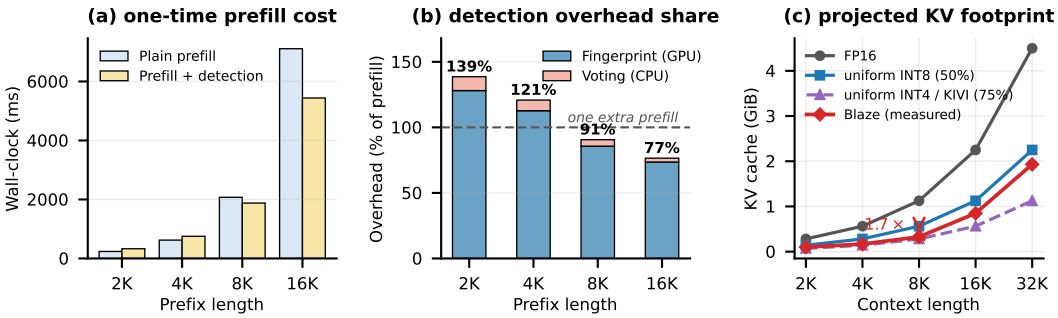

Figure 7: Efficiency analysis on Qwen3-4B. (a) One-time wall-clock cost of plain prefill and of the detection pass (attention fingerprint and voting). (b) Detection cost as a percentage of one plain prefill (bars: attention fingerprint and voting, dashed line: GPU-side work). (c) Projected KV cache footprint under INT8/INT4 storage (36 layers, 8 KV heads, head dim 128).

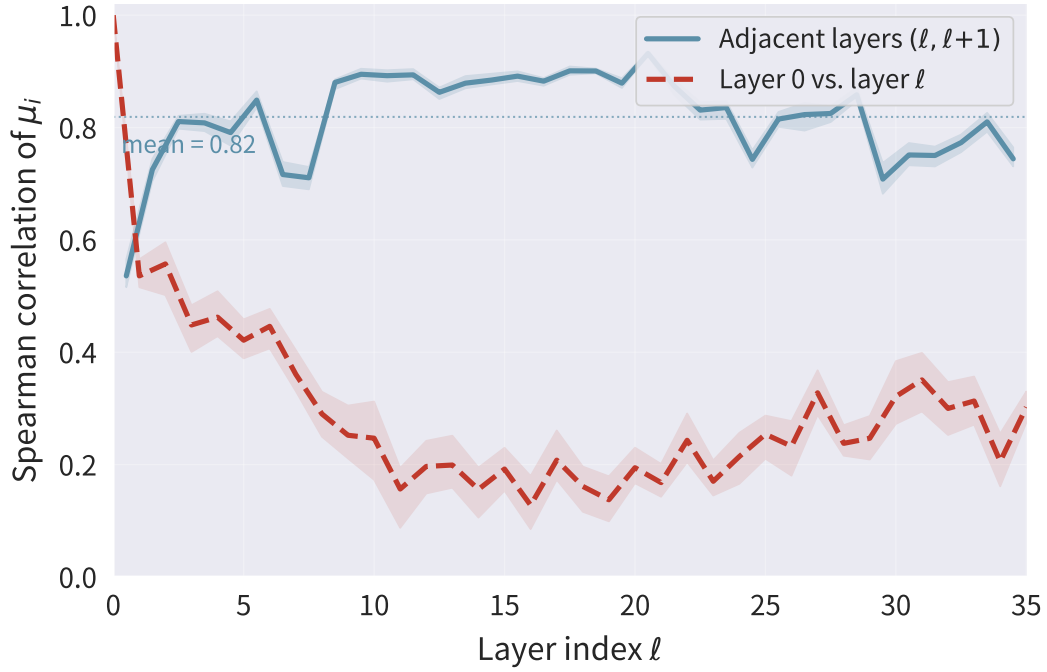

Figure 8: Cross-layer stability of the attention fingerprint μ_i on Qwen3-4B at prefix 4,096. Solid line: adjacent layers $(\ell, \ell+1)$. Dashed line: layer 0 versus layer ℓ. Bands show 10th–90th percentiles.

and on Qwen3-4B and Qwen3-8B its PPL is below FP16 on 82% and 66% of samples, against 28% and 34% for uniform INT4 and 14% and 20% for KIVI, at 70–71% compression against their 75%. On the 4B/7B/8B models the mean paired ΔPPL of Blaze stays between -1.4% and $+0.6\%$ of FP16.

F Additional Tables and Figures

Figure 9 plots the compression–quality trade-off at the 4K prefix, and Figure 5 the compression and paired ΔPPL across prefix lengths.

Table 12: Differences against FP16 and the error of the measurement. (a) Paired ΔPPL (%) against FP16 at the 4K prefix: mean ± standard error of the mean (n=50 samples), positive means higher PPL than FP16, and the second row of each model gives the share of samples with lower PPL than FP16. (b) Error of the measurement for the quantities compared in this paper.

(a) Paired ΔPPL (%) vs. FP16 at 4K					
Qwen3-4B	$+0.25 \pm 0.08$	$+0.13 \pm 0.08$	$+2.42 \pm 0.54$	$+4.78 \pm 0.54$	-1.42 ± 0.25
below FP16 (%)	40%	52%	28%	14%	82%
Qwen3-8B	$+0.11 \pm 0.09$	-0.06 ± 0.10	$+2.50 \pm 0.70$	$+3.53 \pm 0.57$	-0.73 ± 0.25
below FP16 (%)	40%	52%	34%	20%	66%
Llama-3.1-8B	-0.01 ± 0.02	-0.05 ± 0.02	$+2.73 \pm 0.24$	$+2.20 \pm 0.22$	$+0.59 \pm 0.10$
below FP16 (%)	50%	56%	4%	8%	16%
Mistral-7B	$+0.04 \pm 0.01$	$+0.07 \pm 0.02$	$+0.30 \pm 0.16$	$+0.57 \pm 0.16$	$+0.25 \pm 0.08$
below FP16 (%)	38%	26%	46%	30%	38%
Qwen3-30B	$+0.15 \pm 0.09$	$+0.16 \pm 0.10$	$+8.74 \pm 0.63$	$+5.05 \pm 0.62$	$+2.07 \pm 0.26$
below FP16 (%)	44%	34%	0%	0%	12%
(b) Error of the measurement (points)					
PPL, sample standard deviation (ablation and sweep runs)			3.8–4.5		
ΔPPL across samples, standard deviation (Blaze / all methods)			0.6–1.8 / 0.1–4.9		
Accuracy, 300-sample tasks (Table 1)			SE 1.6–2.9		
Accuracy, four-needle cells, n=50 (Table 13)			SE 2.0–5.2		

Table 13: Hard NIAH retrieval (%). Four needles per key, all-needle exact match, 50 samples per cell, 64 generated tokens. Compression in parentheses is computed from the per-block precision assignment of Blaze (Eq. 2). In the last column, PA protection is removed, so no block is forced to INT8 by a PA token. Blaze values at or above FP16 are bold.

Model	Length	FP16	INT8	INT4	**Blaze**	No PA protection
Qwen3-4B	4096	100.0	100.0	92.0	**100.0** (70%)	100.0 (73%)
Qwen3-4B	8192	98.0	98.0	94.0	**98.0** (71%)	92.0 (74%)
Qwen3-4B	16384	98.0	98.0	84.0	**100.0** (62%)	98.0 (75%)
Qwen3-4B	32768	92.0	94.0	84.0	**92.0** (57%)	96.0 (75%)
Llama-3.1-8B	4096	98.0	96.0	98.0	**98.0** (70%)	96.0 (73%)
Llama-3.1-8B	8192	100.0	100.0	98.0	98.0 (71%)	98.0 (74%)
Llama-3.1-8B	16384	100.0	100.0	94.0	98.0 (62%)	96.0 (75%)
Llama-3.1-8B	32768	94.0	96.0	88.0	92.0 (57%)	96.0 (75%)

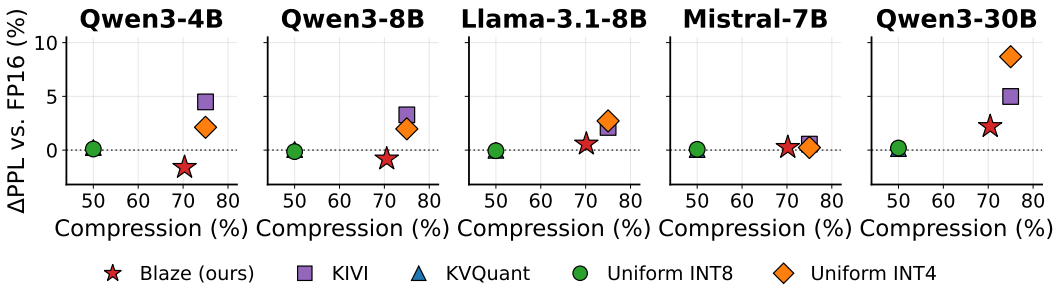

Figure 9: Compression–quality trade-off (prefix 4,096, all methods measured under the same setup). Blaze is shown as red stars.