PN-Mix: Pair-Norm Guided Mixed-Precision Key Cache Quantization for Efficient Long-Output LLM Inference

Gil Jae Song **Sungjoo Yoo***
Seoul National University
jkoi632@snu.ac.kr, sungjoo.yoo@gmail.com

Abstract

Long-output inference in large language models suffers from severe KV cache memory and bandwidth bottlenecks. While KV cache quantization alleviates this issue, low-bit quantization (e.g., 2-bit) severely degrades model accuracy. Recent mixed-precision key-cache quantization methods mitigate this degradation by allocating higher precision to important key channels, but estimate channel saliency using only key statistics. Through attention logit perturbation analysis at the RoPE channel-pair level, we show that quantization errors in channel pairs with large query–key pair-norm products contribute significantly to logit variation. Based on this insight, we propose PN-Mix, a mixed-precision key-cache quantization method for RoPE-based decoder-only LLMs, which allocates higher precision to salient channel pairs identified once during the prefill stage and reused throughout decoding. Across standard reasoning, code-generation, and AIME benchmarks, PN-Mix with 2.26-bit average precision achieves comparable or better average accuracy than prior mixed-precision methods using 2.32–2.39 bits. Furthermore, in our Pareto sweep, PN-Mix degrades gracefully down to 1.5 bits, whereas the compared baselines exhibit sharper performance drops below 2.25 bits.

1 Introduction

Long-output inference has become a central capability of large language models (LLMs), enabling applications that require extended generation such as code generation and complex problem solving. However, long-output decoding exposes a severe systems bottleneck in the key-value (KV) cache, which stores previous keys and values but grows linearly with sequence length and batch size. This growth increases memory pressure, reducing the feasible batch size for serving and limiting the maximum generation length. For example, Qwen3-8B in FP16 runs out of memory on a single A100-40GB GPU when the generation length exceeds 4K at batch size 32, or when the batch size exceeds 20 at an 8K generation length (Figure 6).

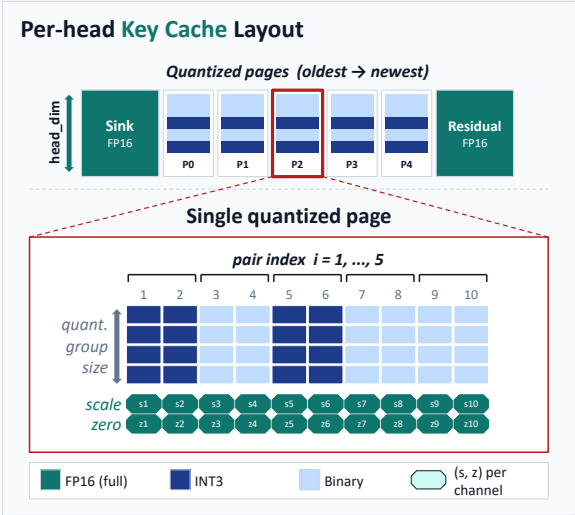

Figure 1: **Post-RoPE key-cache layout of PN-Mix.** Salient RoPE channel pairs selected during prefill are stored in higher precision across quantized pages, while the remaining pairs use lower precision.

KV cache quantization alleviates this bottleneck by storing cached keys and values at lower precision while retaining all token positions and the original cache structure. Yet this compression is inherently lossy: aggressive low-bit quantization (e.g., 2-bit) can introduce errors that substantially degrade reasoning accuracy over long chains of thought (Liu et al., 2025). Thus, effective KV cache quantization requires not only lowering precision, but also allocating the limited bit budget to the most critical components.

Recent mixed-precision methods allocate higher precision to salient key channels, motivated by the greater quantization sensitivity of keys than values (Li et al., 2025; Xia et al., 2026; Hariri et al., 2026). However, their saliency estimates rely on

*Indicates corresponding author

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 28819–28841
October 24-29, 2026 ©2026 Association for Computational Linguistics

key statistics alone and operate at the individual-channel level, overlooking query–key interactions and the channel-pair structure induced by RoPE.

In this work, we propose PN-Mix, a pair-norm guided mixed-precision key-cache quantization method designed for RoPE-based decoder-only LLMs. Our key observation is that key-cache quantization should be guided by its effect on attention logits, rather than by key statistics alone. Through attention-logit perturbation analysis at the RoPE channel-pair level, we show that quantization errors in channel pairs with large query–key pair-norm products can contribute substantially to logit variation. PN-Mix therefore assigns higher precision to such salient RoPE channel pairs. We further find that salient channel pairs estimated from prefill tokens are predictive of full-sequence saliency, enabling PN-Mix to identify them once during prefill and reuse them throughout decoding. As a result, PN-Mix remains input-adaptive without offline calibration and avoids repeated saliency recomputation during decoding.

Overall, this work makes three main contributions:

- We derive a query-aware RoPE channel-pair saliency criterion based on query–key pair-norm products from attention-logit perturbation analysis.
- We propose PN-Mix, a mixed-precision key-cache quantization method that input-adaptively selects salient RoPE channel pairs once during prefill and reuses the resulting selection throughout decoding.
- We show that PN-Mix improves long-output inference by reducing average key-cache precision while preserving accuracy and avoiding decoding-time saliency recomputation.

2 Background

We provide a brief overview of the attention and online KV-cache quantization setup used in this work. For a decoder-layer hidden state $\mathbf{X} \in \mathbb{R}^{T\times D}$ with sequence length T and hidden dimension D, Grouped Query Attention (GQA) (Ainslie et al., 2023) projects $\mathbf{X}$ into $Q \in \mathbb{R}^{H\times T\times d}$ and $K, V \in \mathbb{R}^{H^{kv}\times T\times d}$, where H, H^{kv}, and d denote the number of query heads, KV heads, and the head dimension. RoPE (Su et al., 2024) gives $\tilde{Q}_{h',t} = Q_{h',t}R_t$ and $\tilde{K}_{h,t} = K_{h,t}R_t$, where R_t is a position-dependent block-diagonal rotation over adjacent channel pairs. With causal masking, the attention distribution $A_{h,t}$ and output $O_{h,t}$ at query position t are computed over positions $1, \ldots, t$ as

$$A_{h,t} = \mathrm{softmax}\left(\frac{\tilde{Q}_{h',t}\tilde{K}^{\top}_{g(h'),1:t}}{\sqrt{d}}\right), \qquad O_{h,t} = A_{h,t}V_{g(h'),1:t}. \tag{1}$$

where $g(h')$ maps query head h' to its shared KV head h. During autoregressive decoding, previously computed $\tilde{K}$ and V are stored as the KV cache and reused at later steps. For batch size B and number of layers L, the KV cache memory scales as $\mathcal{O}(BLH^{kv}Td)$, motivating KV-cache quantization for long-output inference.

We adopt asymmetric group quantization as the base quantization scheme. For a group $X \in \mathbb{R}^G$, b-bit asymmetric quantization stores $X_q \in \{0, \ldots, 2^b - 1\}^G$ with scale s_X and zero-point z_X, and reconstructs $\hat{X} = s_X X_q + z_X$:

$$s_X = \frac{\max X - \min X}{2^b - 1}, \quad z_X = \min X, \qquad X_q = \mathrm{clamp}\left(\left\lfloor \frac{X - z_X}{s_X} \right\rceil, 0, 2^b - 1\right). \tag{2}$$

Following common KV-cache quantization practice (Liu et al., 2024), we apply different grouping strategies to keys and values: keys are grouped per channel to handle channel-wise outliers, whereas values are grouped per token to align with the attention weighted-sum operation in Eq. 1. In online decoding, newly generated $\tilde{K}_{h,t}$ and $V_{h,t}$ are kept in an FP16 residual buffer of size N. Once the buffer is full, it is quantized and appended to the quantized cache; during attention, the dequantized cache is concatenated with the FP16 residual buffer.

3 Related Work

Prior work has explored several directions for reducing KV cache memory, including token eviction (Zhang et al., 2023; Li et al., 2024; Cai et al., 2025), low-rank approximation (Chang et al., 2025; Lee et al., 2025), merging (Zhang et al., 2024b; Wang et al., 2024), pruning (Xu et al., 2025; Zhang et al., 2025), and quantization. Among them, KV cache quantization reduces memory by lowering the representation precision of cached tensors while retaining the cache structure. We focus on KV cache quantization, which can be broadly divided into uniform-precision and mixed-precision approaches.

3.1 Uniform-Precision Quantization

Uniform-precision KV cache quantization assigns the same bit-width to all quantized cache elements. Prior methods reduce low-bit quantization error through key/value-specific grouping (Liu et al., 2024), distribution-aware quantization (Hooper et al., 2024), residual correction (Kang et al., 2024), outlier-token preservation (Su et al., 2025a), Walsh–Hadamard channel rotations (Su et al., 2025b), projection-based 1-bit sign quantization (Zandieh et al., 2025), and polar-coordinate quantization of RoPE channel pairs (Wu et al., 2025). However, because they assign the same bit-width regardless of cache-element sensitivity, uniform-precision methods may insufficiently control errors in important components, which can accumulate over long chains of thought in reasoning models (Liu et al., 2025).

3.2 Mixed-Precision Quantization

Mixed-precision quantization instead allocates a limited bit budget according to estimated importance. Existing methods differ in allocation granularity, including token-level (Yang et al., 2024), layer-level (Li et al., 2025; Liu et al., 2026), and channel-level allocation. Since key-cache quantization importance is highly non-uniform across channels due to channel-wise outliers (Liu et al., 2024; Hooper et al., 2024), recent channel-level methods focus on assigning higher precision to salient key channels. ChanMix (Liao and Wen, 2026) uses an offline calibration set to identify retrieval-head channels, assigns them 4-bit precision, and allocates 3/2/1-bit precision to the remaining channels via min–max range-based clustering. Kitty (Xia et al., 2026) dynamically re-estimates channel saliency for each quantization group using average key magnitude, assigning 4-bit precision to salient channels and 2-bit precision to the rest.

Although these methods improve over uniform precision, they estimate saliency from $\tilde{K}$ alone and assign precision at the individual-channel level. This misses two structures that directly affect attention: attention logits are determined by the interaction between $\tilde{Q}$ and $\tilde{K}$, and RoPE couples channels into pairs. In contrast, PN-Mix estimates saliency from query–key interactions at the RoPE channel-pair level and allocates mixed precision to the key cache accordingly.

4 Preliminary Analysis

To achieve accuracy-preserving ∼2.25-bit KV-cache compression for long-output inference, we investigate three design questions: (1) which cache (key or value) to refine; (2) how to define an importance score for assigning higher precision within the key cache; and (3) how to exploit this importance score efficiently during long-output inference. We examine these questions sequentially and use the resulting observations to guide our mixed-precision key-cache quantization design.

4.1 Sensitivity of Key and Value Caches to Quantization

Under a 2.25-bit budget[1], prior work indicates the key cache is more sensitive to quantization than the value cache (Li et al., 2025; Xia et al., 2026). We verify this by quantizing either $\tilde{K}$ or V to 2.25 bits and comparing against an FP16 baseline on AIME datasets. We evaluate DeepSeek-R1-Distill-Llama-8B (R1-Llama-8B) and Qwen3-8B, generating up to 32K tokens per response. Accounting for attention sinks (Xiao et al., 2024), we also evaluate variants preserving the initial 32 tokens in FP16, following prior work (Xia et al., 2026).

Table 8 shows that V-only quantization causes less degradation than $\tilde{K}$-only quantization, even without sink-token protection. While preserving the first 32 tokens in FP16 improves both, V-only quantization approaches the FP16 baseline whereas $\tilde{K}$-only quantization still exhibits non-negligible degradation.

These results lead to two design choices. First, sink tokens should be preserved in full precision for both caches. Second, key-cache quantization should be refined more carefully than value-cache quantization under low-bit compression.

4.2 Designing an Attention-Aware Importance Score

Having identified the key cache as the more sensitive component under low-bit quantization, we next consider how to determine which parts of the key cache should receive higher precision. Existing mixed-precision key-cache quantization methods estimate channel importance using statistics derived only from the key cache. ChanMix (Liao and Wen, 2026) uses the channel range, $\max_t \tilde{K}_{h,t,c} -$

[1] Effective bit-width for 2-bit asymmetric quantization ($G = 128$) with per-group FP16 scales and zero-points. Tokens in the FP16 residual buffer are excluded.

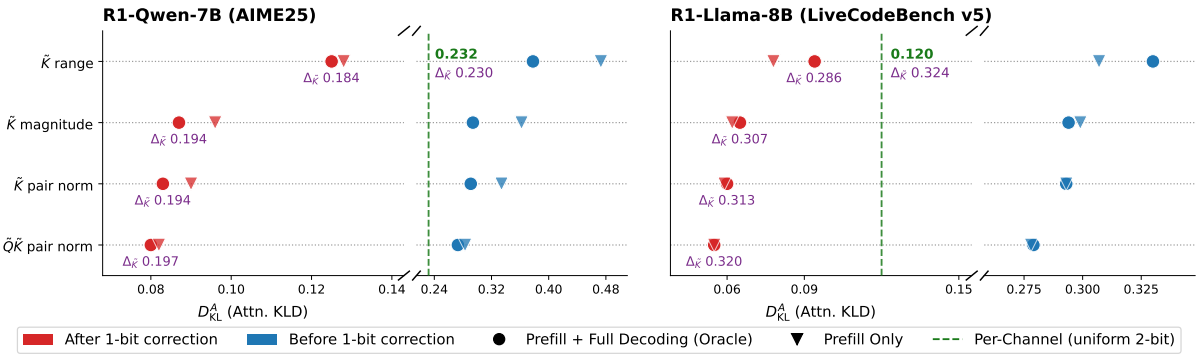

Figure 2: **Simulation results under H3(50%)L1 key-cache bit allocation.** Each marker shows mean D^A_{KL}; adjacent purple values show mean $\Delta_{\tilde{K}}$. For each criterion, we pseudo-quantize $\tilde{K}$ with midpoint correction for 1-bit groups (Eq. 7); $\Delta_{\tilde{K}}$ is measured on the reconstructed $\tilde{K}'$, while D^A_{KL} is computed from $A'_{h,t}$ recomputed using the corresponding $\tilde{Q}$.

$\min_t \tilde{K}_{h,t,c}$, whereas Kitty (Xia et al., 2026) uses the average channel magnitude, $\mathbb{E}_t[|\tilde{K}_{h,t,c}|]$.

However, these key-only, channel-wise criteria do not explicitly account for how the key cache is used in attention. After RoPE, adjacent key channels are coupled into channel pairs, and each key contributes to the attention logits through its interaction with the query. Motivated by these observations, we decompose the attention-logit perturbation caused by post-RoPE key quantization at the RoPE channel-pair level.

For a fixed head h, we define the i-th query and key channel-pair vectors at query position t and key position s as $\mathbf{q}_i^t := (\tilde{Q}_{h,t,2i-1}, \tilde{Q}_{h,t,2i})$ and $\mathbf{k}_i^s := (\tilde{K}_{h,s,2i-1}, \tilde{K}_{h,s,2i})$. For $s \leq t$, dropping the common scaling factor $1/\sqrt{d}$, the corresponding attention logit can be decomposed as

$$\tilde{Q}_{h,t}\tilde{K}_{h,s}^\top = \sum_{i=1}^{d/2} \|\mathbf{q}_i^t\|_2 \|\mathbf{k}_i^s\|_2 \cos(\phi_i^{t,s}) \quad (3)$$

where $\phi_i^{t,s}$ denotes the angle between $\mathbf{q}_i^t$ and $\mathbf{k}_i^s$. Let $\mathbf{k}_i'^s$ denote the reconstructed key pair after quantization. We express the resulting changes in the key-pair-norm and angular term as $\|\mathbf{k}_i'^s\|_2 = \|\mathbf{k}_i^s\|_2 + \epsilon_i^s$ and $\cos(\phi_i'^{t,s}) = \cos(\phi_i^{t,s}) + \delta_i^{t,s}$, respectively, and define the relative perturbation of the key-pair-norm as $\rho_i^s = \epsilon_i^s / \|\mathbf{k}_i^s\|_2$. The resulting difference between the original and quantized attention logits is

$$\begin{aligned} &\tilde{Q}_{h,t}\tilde{K}_{h,s}^\top - \tilde{Q}_{h,t}\tilde{K}'_{h,s}{}^\top \\ &= -\sum_{i=1}^{d/2} \|\mathbf{q}_i^t\|_2 \|\mathbf{k}_i^s\|_2 \left(\delta_i^{t,s} + \cos\phi_i^{t,s}\rho_i^s + \rho_i^s\delta_i^{t,s}\right) \end{aligned} \quad (4)$$

In Eq. 4, the term in parentheses captures the perturbation induced by quantization, while the query–key pair-norm product scales its contribution to the attention-logit error. The exact perturbation terms, such as ρ_i^s and $\delta_i^{t,s}$, depend on the reconstructed key and are therefore not available when the bit-width allocation is determined. We therefore use the query–key pair-norm product, which is directly available from the unquantized attention states, as an attention-aware proxy for the pre-quantization sensitivity of each RoPE channel pair.

Using this quantity directly as an importance score would require aggregating $\|\mathbf{q}_i^t\|_2\|\mathbf{k}_i^s\|_2$ over causal query–key position pairs (t, s) with $s \leq t$. Since our goal is to rank channel pairs for precision allocation rather than to reproduce the causal-joint score exactly, we use a separable approximation that summarizes the query and key pair-norms independently over the observed tokens. This avoids retaining the explicit causal pairing structure while preserving a compact score for channel-pair ranking. Specifically, we define the importance score for channel pair i in head h as

$$\sigma_{h,i} = \bar{q}_{h,i} \cdot \bar{k}_{h,i} = \mathbb{E}_t\left[\|\mathbf{q}_i^t\|_2\right] \cdot \mathbb{E}_s[\|\mathbf{k}_i^s\|_2]\,. \quad (5)$$

This separable form does not explicitly preserve the causal pairing between query and key positions.

To assess the approximation, we compare it against a causal-joint score obtained by averaging $\|\mathbf{q}_i^t\|_2\|\mathbf{k}_i^s\|_2$ over all causal pairs (t, s) with $s \leq t$. As shown in Table 1, Spearman rank correlation and top-k% selection overlap indicate that the separable formulation largely preserves the salient-pair ordering of the causal-joint score while avoiding explicit computation over all causal query–key pairs.

Model / Dataset	Spearman ρ	Top-10% overlap	Top-50% overlap
R1-Llama-8B / LiveCodeBench v5	$0.996_{\pm 0.000}$	$96.2_{\pm 0.3}$	$97.9_{\pm 0.1}$
R1-Qwen-7B / AIME25	$0.990_{\pm 0.003}$	$95.3_{\pm 0.6}$	$96.2_{\pm 0.5}$

Table 1: **Separable vs. causal-joint pair-norm score.** The table reports the similarity in ranking and selection between $\sigma_{h,i}$ (Eq. 5) and the causal-joint score computed over prefill states. Metrics are averaged over layers and KV heads and reported as mean$_{\pm \mathrm{std}}$ across samples.

Next, we evaluate whether the proposed importance score leads to more accuracy-preserving key-cache quantization than existing criteria. The evaluation uses post-RoPE $\tilde{Q}$ and $\tilde{K}$ states collected from R1-Qwen-7B on AIME25 and R1-Llama-8B on LiveCodeBench v5 (Jain et al., 2025). Quantization quality is measured by two metrics: the key-cache reconstruction error, $\Delta_{\tilde{K}} := \frac{1}{H^{kv}Td}\|\tilde{K} - \tilde{K}'\|_F^2$, where $\tilde{K}'$ is the reconstructed key cache, and the attention-distribution KL divergence, $D_{\mathrm{KL}}^{A} := \mathbb{E}_{h,t}[D_{\mathrm{KL}}(A_{h,t} \,\|\, A'_{h,t})]$, where $A'_{h,t}$ is recomputed using $\tilde{K}'$.

For comparison, we consider uniform 2-bit per-channel quantization and a mixed-precision setting, denoted H3(50%)L1, that assigns 3 bits to the top 50% of channels or channel pairs and 1 bit to the remainder. Bit allocation is determined by four importance criteria: the channel-wise $\tilde{K}$ range, the channel-wise $\tilde{K}$ magnitude, the $\tilde{K}$ pair norm $\bar{k}_{h,i}$, and the proposed $\tilde{Q}\tilde{K}$ pair norm $\sigma_{h,i}$. For the first two criteria, individual channels are ranked by importance, whereas the latter two rank channel pairs. In all cases, salient channels or channel pairs are selected using full-sequence oracle statistics. All methods use asymmetric per-channel quantization with group size $G = 128$ (Eq. 2) under the same effective ~2.25-bit budget. Figure 2 summarizes the results, with full simulation details provided in Appendix B.

As shown in Figure 2, the proposed $\tilde{Q}\tilde{K}$ pair norm achieves the lowest D_{KL}^{A} among all compared mixed-precision criteria on both models, indicating better preservation of the attention distribution under the same effective bit budget. Its improvement over the $\tilde{K}$ pair norm further indicates that incorporating query-side information provides additional benefit beyond accounting for the RoPE channel-pair structure alone. Based on these results, we use $\sigma_{h,i}$ in Eq. 5 as the importance score for mixed-precision key-cache quantization.

4.3 Efficient Importance Estimation for Long-Output Inference

With $\sigma_{h,i}$ established as the importance score, the remaining challenge is to estimate it efficiently during long-output inference. ChanMix relies on offline profiling, eliminating online selection overhead but sacrificing input-adaptive channel allocation, whereas Kitty performs input-adaptive selection for each quantization group. However, this requires repeatedly identifying salient channels and storing a separate set of channel indices for each group. As decoding proceeds and more groups accumulate, the resulting index storage and loading overhead can grow substantially.

To retain input-adaptive importance while avoiding repeated salient-pair selection and index management during long decoding, we choose to compute $\sigma_{h,i}$ once from the prefill tokens and reuse the selected channel pairs throughout decoding.

For this prefill-only strategy to be effective, the prefill-based importance score should satisfy two properties. First, it should be predictive: channel pairs identified as important during prefill should remain salient throughout subsequent decoding. Second, it should be selective: the importance should be concentrated on a subset of channel pairs so that assigning higher precision to the selected pairs remains effective under a limited bit budget.

We first evaluate the predictiveness of prefill-based selection by comparing its quantization quality against a full-sequence oracle that computes importance scores using the complete sequence. Using the same H3(50%)L1 simulation setup as in §4.2, Figure 2 shows that the prefill-based $\tilde{Q}\tilde{K}$ pair norm achieves D_{KL}^{A} close to its full-sequence counterpart on both evaluated models.

To further understand why prefill-based selection remains effective, Appendix C compares the channel pairs identified from prefill with those that are salient during decoding. Among the evaluated criteria, the prefill-based $\tilde{Q}\tilde{K}$ pair norm yields the highest average Jaccard overlap with decoding-time salient pairs. Moreover, Appendix L shows that this overlap does not systematically decrease as decoding progresses, including in long-generation and multi-topic settings.

We next examine selectivity. Approximately 97–99% of the total importance is concentrated in the top 50% of channel pairs (Figure 5); these values

denote the median across layers, KV heads, and prompts.

Together, these results indicate that $\sigma_{h,i}$ can be computed once during the input prefill stage and that the selected channel pairs can be reused throughout long-output decoding.

5 Methodology

This section presents PN-Mix, which integrates the channel-pair importance score $\sigma_{h,i}$ from §4.2 into the online KV-cache quantization setup in §2. As shown in Figure 1, PN-Mix assigns mixed precision to key-cache channel pairs according to $\sigma_{h,i}$ while keeping sink tokens and the residual buffer in FP16.

5.1 Prefill-Stage Salient Channel-Pair Prediction

During prefill, PN-Mix computes $\sigma_{h,i}$ from the prefill $\tilde{Q}$ and $\tilde{K}$ states for each layer, KV head h, and channel pair i. Under GQA, each KV head is shared by multiple query heads, so we average the query-side scores over the query heads sharing h:

$$\begin{aligned} \bar{q}_{h',i} &= \mathbb{E}_t\left[\left\|(\tilde{Q}_{h',t,2i-1},\, \tilde{Q}_{h',t,2i})\right\|_2\right], \\ \bar{k}_{h,i} &= \mathbb{E}_t\left[\left\|(\tilde{K}_{h,t,2i-1},\, \tilde{K}_{h,t,2i})\right\|_2\right], \\ \sigma_{h,i} &= \frac{1}{|g^{-1}(h)|} \sum_{h' \in g^{-1}(h)} \bar{q}_{h',i}\bar{k}_{h,i}, \end{aligned} \tag{6}$$

where $g(h')$ maps query head h' to its shared KV head. For each KV head h, $\mathcal{S}_h$ contains the top-$r\%$ channel pairs by $\sigma_{h,i}$. For a batch of size B, we store the selected pairs as a binary mask of size $B \times L \times H^{kv} \times (d/2)$ bits.

5.2 Online KV-Cache Quantization Scheme

During decoding, PN-Mix keeps the first N_s sink tokens and the residual buffer of size N in FP16 for both keys and values. When the buffer is full, the buffered keys are quantized with per-channel asymmetric groups: channel pairs in $\mathcal{S}_h$ use b_H bits, and the remaining pairs use b_L bits. For 1-bit groups, we apply the correction in Eq. 7 (Jie et al., 2025). This shifts the two reconstruction levels from the endpoints toward the bin midpoints, reducing the bias of extreme 1-bit min–max quantization. Motivated by the lower quantization sensitivity of value caches observed in §4.1, PN-Mix applies per-token asymmetric group quantization to values at b_V bits.

$$s_X = \frac{\max X - \min X}{2}, \quad z_X = \frac{3\min X + \max X}{4}. \tag{7}$$

6 Experiments

6.1 Experimental Setup

Models and tasks. We evaluate accuracy with lm-evaluation-harness (Gao et al., 2023). For standard reasoning benchmarks, we use Llama-3.1-8B-Instruct (Grattafiori et al., 2024), Qwen3-8B/32B (Yang et al., 2025), and Qwen2.5-14B-Instruct (Qwen et al., 2025) on GSM8K (Cobbe et al., 2021) (8-shot), MATH-Algebra (Hendrycks et al., 2021) (4-shot), HumanEval (Chen et al., 2021), and GPQA-Diamond (Rein et al., 2024) (5-shot), with a 4096-token generation limit. For AIME 2024 (Zhang and Math-AI, 2024) and AIME 2025 (Zhang and Math-AI, 2025), we use DeepSeek-R1-Distill-Llama-8B (Guo et al., 2025), Qwen3-8B, and DeepSeek-R1-Distill-Qwen-7B (Guo et al., 2025) with $(\text{temperature}, \text{top-}p, \text{top-}k) = (0.6, 0.95, 20)$ and a 32768-token limit. We report mean and standard deviation over five runs for standard benchmarks and 10 responses per question for AIME; for HumanEval, we report pass@1 based on unit-test correctness, and all other tasks use extracted-answer accuracy.

Baselines. We compare PN-Mix with uniform-precision baselines, KIVI (Liu et al., 2024) and PolarQuant (Wu et al., 2025), and mixed-precision baselines, ChanMix (Liao and Wen, 2026) and Kitty (Xia et al., 2026). Unless otherwise specified, all baselines use group size $G = 128$ and residual buffer length $N = 128$.

For models without official ChanMix profiles, we use reproduced profiles following the original profiling procedure; details of the reproduction and validation are provided in Appendix H.

PN-Mix configurations. PN-Mix uses $N = 128$, $b_H = 3$, $b_L = 1$, and $N_s = 32$, following the baseline residual length and Kitty's sink-token setting. We report two similar-budget operating points: PN-Mix (1) ($r = 50\%$, $G = 128$, 2.26 bits) and PN-Mix (2) ($r = 40\%$, $G = 64$, 2.29 bits), comparing a larger high-bit ratio with a smaller group size.

Hardware and software. Accuracy experiments use NVIDIA RTX A6000 GPUs, and inference-efficiency experiments use a single NVIDIA A100-SXM4-40GB GPU. All experiments use PyTorch 2.4.1, CUDA 12.1, and FlashAttention (Dao et al., 2022) 2.7.4.post1.

Model	Method	K bits	GSM8K	MATH-Alg.	HumanEval	GPQA-Dia	Avg.
Llama3.1-8B-Instruct	FP16	16	$76.83_{\pm 0.90}$	$48.17_{\pm 0.89}$	$64.51_{\pm 1.70}$	$26.63_{\pm 1.22}$	54.04
	KIVI	2.25	$66.97_{\pm 1.07}$	$35.27_{\pm 0.80}$	$55.49_{\pm 3.18}$	$23.67_{\pm 2.06}$	45.35
	PolarQuant	2.25	$26.85_{\pm 0.49}$	$10.38_{\pm 0.85}$	$34.27_{\pm 1.30}$	$20.41_{\pm 2.35}$	22.98
	ChanMix	2.32	$70.51_{\pm 0.75}$	$37.57_{\pm 0.70}$	$59.63_{\pm 2.23}$	$23.37_{\pm 1.38}$	47.77
	Kitty	2.39	$74.51_{\pm 0.62}$	$46.25_{\pm 0.72}$	$62.68_{\pm 1.78}$	$23.98_{\pm 2.14}$	51.86
	PN-Mix (1)	2.26	$\mathbf{75.66}_{\pm 0.94}$	$\mathbf{46.57}_{\pm 1.43}$	$\mathbf{64.39}_{\pm 2.10}$	$25.71_{\pm 2.59}$	**53.08**
	PN-Mix (2)	2.29	$75.65_{\pm 0.61}$	$46.28_{\pm 1.24}$	$63.78_{\pm 2.30}$	$\mathbf{26.33}_{\pm 3.02}$	53.01
Qwen3-8B	FP16	16	$94.27_{\pm 0.64}$	$88.41_{\pm 0.90}$	$85.85_{\pm 0.46}$	$42.86_{\pm 0.56}$	77.85
	KIVI	2.25	$88.20_{\pm 0.74}$	$69.47_{\pm 0.23}$	$79.88_{\pm 2.18}$	$35.10_{\pm 0.59}$	68.16
	PolarQuant	2.25	$1.15_{\pm 0.09}$	$0.00_{\pm 0.00}$	$2.07_{\pm 0.30}$	$0.82_{\pm 0.69}$	1.01
	ChanMix †	2.32	$90.34_{\pm 5.86}$	$86.57_{\pm 4.01}$	$83.65_{\pm 1.30}$	$38.97_{\pm 0.76}$	74.88
	Kitty	2.39	$91.93_{\pm 0.44}$	$82.29_{\pm 0.68}$	$83.90_{\pm 1.62}$	$39.59_{\pm 2.53}$	74.43
	PN-Mix (1)	2.26	$93.31_{\pm 0.31}$	$\mathbf{88.76}_{\pm 0.38}$	$\mathbf{86.34}_{\pm 0.83}$	$40.41_{\pm 2.20}$	**77.21**
	PN-Mix (2)	2.29	$\mathbf{93.89}_{\pm 0.25}$	$87.40_{\pm 0.46}$	$84.87_{\pm 2.26}$	$\mathbf{41.84}_{\pm 2.21}$	76.66
Qwen2.5-14B-Instruct	FP16	16	$90.83_{\pm 0.29}$	$78.16_{\pm 0.29}$	$77.07_{\pm 2.46}$	$40.00_{\pm 3.27}$	71.52
	KIVI	2.25	$85.66_{\pm 1.07}$	$72.00_{\pm 1.26}$	$71.71_{\pm 2.43}$	$36.73_{\pm 0.72}$	66.52
	PolarQuant	2.25	$61.47_{\pm 1.18}$	$44.53_{\pm 1.04}$	$58.17_{\pm 1.71}$	$27.76_{\pm 2.77}$	47.98
	ChanMix	2.32	$88.79_{\pm 0.53}$	$73.95_{\pm 0.66}$	$74.88_{\pm 2.95}$	$\mathbf{41.02}_{\pm 3.19}$	69.66
	Kitty	2.39	$\mathbf{90.46}_{\pm 0.79}$	$\mathbf{80.39}_{\pm 0.93}$	$77.20_{\pm 3.38}$	$40.00_{\pm 2.38}$	72.01
	PN-Mix (1)	2.26	$90.07_{\pm 0.55}$	$79.58_{\pm 0.87}$	$\mathbf{78.17}_{\pm 2.13}$	$40.61_{\pm 2.51}$	**72.11**
	PN-Mix (2)	2.29	$89.01_{\pm 0.46}$	$78.15_{\pm 1.54}$	$77.44_{\pm 1.02}$	$40.71_{\pm 1.81}$	71.33
Qwen3-32B	FP16	16	$90.80_{\pm 0.42}$	$84.62_{\pm 1.00}$	$85.61_{\pm 1.13}$	$53.27_{\pm 2.20}$	78.58
	KIVI	2.25	$87.47_{\pm 0.62}$	$59.87_{\pm 1.15}$	$84.51_{\pm 0.62}$	$46.73_{\pm 1.53}$	69.65
	PolarQuant	2.25	$63.21_{\pm 0.64}$	$1.31_{\pm 0.13}$	$46.71_{\pm 2.27}$	$7.55_{\pm 1.63}$	29.70
	ChanMix †	2.32	$90.08_{\pm 0.84}$	$82.73_{\pm 0.78}$	$86.46_{\pm 2.36}$	$48.78_{\pm 2.15}$	77.01
	Kitty	2.39	$\mathbf{91.10}_{\pm 0.54}$	$76.73_{\pm 0.98}$	$87.68_{\pm 2.36}$	$47.86_{\pm 1.53}$	75.84
	PN-Mix (1)	2.26	$90.33_{\pm 0.30}$	$82.81_{\pm 0.60}$	$\mathbf{88.29}_{\pm 1.36}$	$\mathbf{50.82}_{\pm 1.39}$	**78.06**
	PN-Mix (2)	2.29	$90.61_{\pm 0.41}$	$\mathbf{83.91}_{\pm 0.71}$	$85.37_{\pm 1.77}$	$48.88_{\pm 0.59}$	77.19

Table 2: **Accuracy comparison on standard reasoning and code-generation benchmarks.** All baselines are configured to use an average key-cache precision in the 2.25–2.4-bit range, and the value cache is kept in FP16. Each cell reports the mean ± standard deviation over five runs. Bold indicates the best non-FP16 result in each column. † denotes the results obtained by applying the reproduced profiling procedure and the resulting profile to the model.

6.2 Downstream Task Accuracy

We evaluate PN-Mix accuracy in two settings: (i) standard reasoning benchmarks on general-purpose models (Table 2) and (ii) AIME benchmarks on reasoning models (Table 3).

All baselines are configured to use 2.25–2.4 average key-cache bits; for channel-level mixed-precision methods, we keep their original bit-width design and adjust only the high-precision channel ratio. These tables are intended as comparisons under comparable low key-cache bit budgets rather than best-effort evaluations at each method's recommended configuration. The exact average bit calculation and method-specific settings are provided in Appendix H. To isolate key-cache quantization, Table 2 and the b_V=16 columns of Table 3 keep values in full precision. For the b_V=2 columns of Table 3, we follow each method's original value-cache quantization design to evaluate full KV-cache compression.

Standard reasoning and code-generation benchmarks on general-purpose models. Table 2 highlights the challenge of uniform low-bit quantization. Uniform 2-bit KIVI incurs a 5.0–9.7 point drop from FP16 in the Avg. column across all four models. In contrast, PN-Mix (1), using an average key-cache precision of only 2.26 bits, remains within one point of FP16 in Avg. across all four models. More importantly, despite using fewer bits than ChanMix and Kitty, PN-Mix (1) achieves comparable or higher Avg. scores across the evaluated Llama and Qwen models spanning 8B to 32B parameters. At the task level, the differences from Kitty on GSM8K are relatively small, while PN-Mix (1) tends to achieve comparable or better results on HumanEval and GPQA-Diamond. Between the two PN-Mix operating points, PN-Mix (1) yields a higher Avg. score for all four models while using slightly fewer bits than PN-Mix (2).

AIME benchmarks on reasoning models. Table 3 shows how effectively PN-Mix preserves AIME24 and AIME25 accuracy on reasoning models under a low key-cache bit budget. Despite using

Model	Method	K bits	AIME24		AIME25	
			b_V=16	b_V=2	b_V=16	b_V=2
R1-Llama-8B	FP16	16	$48.33_{\pm 6.19}$		$31.33_{\pm 8.19}$	
	KIVI	2.25	$28.67_{\pm 7.02}$	$12.00_{\pm 4.99}$	$26.67_{\pm 5.16}$	$19.33_{\pm 3.59}$
	PolarQuant	2.25	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
	ChanMix	2.32	$28.00_{\pm 6.70}$	$19.00_{\pm 5.59}$	$26.00_{\pm 3.27}$	$16.67_{\pm 4.47}$
	Kitty	2.39	$42.00_{\pm 5.62}$	$42.00_{\pm 5.42}$	$35.00_{\pm 6.19}$	$30.00_{\pm 5.96}$
	PN-Mix (1)	2.26	$48.33_{\pm 5.43}$	$\mathbf{47.67}_{\pm 6.16}$	$29.67_{\pm 7.22}$	$31.00_{\pm 4.23}$
	PN-Mix (2)	2.29	$\mathbf{50.67}_{\pm 5.54}$	$44.33_{\pm 5.97}$	$\mathbf{36.00}_{\pm 4.16}$	$\mathbf{33.00}_{\pm 5.04}$
Qwen3-8B	FP16	16	$76.33_{\pm 4.82}$		$67.00_{\pm 4.58}$	
	KIVI	2.25	$64.00_{\pm 5.54}$	$56.33_{\pm 4.33}$	$55.33_{\pm 4.99}$	$51.33_{\pm 3.06}$
	PolarQuant	2.25	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
	ChanMix†	2.32	$69.66_{\pm 4.06}$	$60.66_{\pm 7.11}$	$61.00_{\pm 4.48}$	$57.00_{\pm 4.33}$
	Kitty	2.39	$70.67_{\pm 5.93}$	$71.33_{\pm 4.27}$	$63.33_{\pm 5.16}$	$62.00_{\pm 5.62}$
	PN-Mix (1)	2.26	$74.67_{\pm 3.06}$	$\mathbf{74.67}_{\pm 6.00}$	$65.33_{\pm 4.99}$	$64.00_{\pm 6.63}$
	PN-Mix (2)	2.29	$\mathbf{76.00}_{\pm 4.67}$	$74.33_{\pm 3.67}$	$\mathbf{68.00}_{\pm 4.99}$	$\mathbf{65.33}_{\pm 2.67}$
R1-Qwen-7B	FP16	16	$54.33_{\pm 5.39}$		$39.33_{\pm 7.57}$	
	KIVI	2.25	$35.33_{\pm 4.76}$	$25.00_{\pm 7.64}$	$25.00_{\pm 4.53}$	$21.00_{\pm 4.23}$
	PolarQuant	2.25	$8.67_{\pm 4.52}$	$9.00_{\pm 3.96}$	$9.67_{\pm 3.48}$	$7.00_{\pm 2.77}$
	ChanMix†	2.32	$45.33_{\pm 2.67}$	$46.33_{\pm 6.40}$	$\mathbf{38.66}_{\pm 5.61}$	$34.66_{\pm 4.26}$
	Kitty	2.39	$48.67_{\pm 7.18}$	$45.67_{\pm 5.59}$	$37.33_{\pm 4.42}$	$\mathbf{36.33}_{\pm 3.78}$
	PN-Mix (1)	2.26	$\mathbf{49.33}_{\pm 5.73}$	$\mathbf{47.67}_{\pm 3.96}$	$36.33_{\pm 4.58}$	$35.00_{\pm 5.82}$
	PN-Mix (2)	2.29	$48.67_{\pm 5.81}$	$45.67_{\pm 5.39}$	$36.67_{\pm 6.83}$	$32.67_{\pm 5.54}$

Table 3: **Accuracy comparison on AIME.** For each model, we report both settings where the value cache is kept in FP16 (b_V=16) and where it is quantized to the 2-bit range (b_V=2). Each cell reports the mean $\pm$ standard deviation over 10 responses per question. Bold indicates the best non-FP16 result in each column. † denotes the results obtained by applying the reproduced profiling procedure and the resulting profile to the model.

Importance score	1-bit	sink	AIME24	HumanEval
FP16	–	–	$76.33_{\pm 4.82}$	$85.85_{\pm 0.46}$
$\tilde{Q}\tilde{K}$ pair norm	✓	✓	$76.00_{\pm 4.67}$	$84.87_{\pm 2.26}$
$\tilde{Q}\tilde{K}$ pair norm	✓	×	$72.00_{\pm 5.42}$	$84.02_{\pm 2.54}$
$\tilde{Q}\tilde{K}$ pair norm	×	✓	$69.67_{\pm 6.90}$	$80.36_{\pm 2.48}$
$\tilde{K}$ range	✓	✓	$70.67_{\pm 3.59}$	$83.78_{\pm 2.48}$
$\tilde{K}$ magnitude	✓	✓	$73.00_{\pm 5.04}$	$84.39_{\pm 1.95}$
$\tilde{K}$ pair norm	✓	✓	$74.67_{\pm 4.00}$	$84.63_{\pm 2.01}$
$\tilde{Q}\tilde{K}$ pair norm*	✓	✓	$72.33_{\pm 5.39}$	$82.31_{\pm 0.01}$

Table 4: **Ablation of PN-Mix design choices on Qwen3-8B.** All variants use PN-Mix (2) with FP16 values; the highlighted row is the final configuration and (*) denotes the online variant.

fewer key bits than ChanMix and Kitty, PN-Mix records the highest mean accuracy among the non-FP16 methods in five of the six model–benchmark settings, with R1-Qwen-7B on AIME25 being the exception. This pattern is also observed when the value cache is quantized to 2 bits.

6.3 Ablation Study

We further validate PN-Mix with two analyses, keeping the value cache in full precision to isolate key-cache quantization effects.

Design component ablation. Table 4 summarizes the component ablations. Removing either sink-token preservation or the 1-bit correction degrades accuracy, with the latter causing a larger drop, indicating that both components are important for stable low-bit key-cache quantization. Replacing the proposed $\tilde{Q}\tilde{K}$ pair norm with $\tilde{K}$-only criteria lowers mean accuracy, especially on AIME24, supporting the use of query-side information. The $\tilde{Q}\tilde{K}$ pair norm* row tests online reselection, which accumulates pair-norm statistics during decoding and reselects salient pairs for each quantization group. This online variant not only underperforms PN-Mix but also requires storing and loading group-specific salient channel-pair indices, indicating that prefill-based salient channel-pair selection is more effective and practical under the evaluated conditions.

Bit-rate Pareto frontier. While the main results focus near 2.25 average key-cache bits, we also evaluate AIME24 accuracy on Qwen3-8B and R1-Llama-8B at target bit-widths of 3.0, 2.5, 2.25, 2.0, and 1.5 bits. For each method, we sweep applicable b_H, b_L, G, r settings and select the closest configuration to each target; details are in Appendix I. For Kitty, we keep its original $(b_H, b_L) = (4, 2)$ design for $\geq$2.25 bits and additionally sweep b_H, b_L below 2.25 bits. Figure 3 shows that although both Kitty and PN-Mix are close to FP16 at 2.5–3.0 bits, PN-

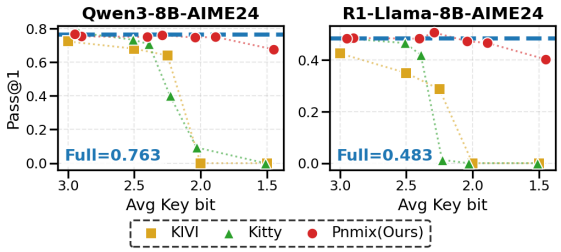

Figure 3: **Pareto frontier in the 1.5–3.0-bit range**. Detailed configurations and model accuracies are provided in Appendix I.

Mix degrades more gracefully at lower bit-widths, indicating that its advantage extends beyond the 2.25-bit operating point.

6.4 Inference Efficiency

We compare PN-Mix with FP16 and Kitty in terms of memory scaling (Figure 6) and end-to-end latency (Table 5) on Qwen3-8B using a single NVIDIA A100-SXM4-40GB GPU. Kitty[2] is evaluated using its official implementation with the original $(b_H, b_L) = (4, 2)$, $r = 25\%$ setting. For PN-Mix, we use our Triton (Tillet et al., 2019) kernel implementation (Appendix G) for the lower-bit PN-Mix (1) setting, with $b_H = 3$, $b_L = 1$, and $b_V = 2$. In all efficiency experiments, each method preallocates KV-cache memory for `max_cache_len` tokens before running inference.

Memory scaling and throughput. We evaluate the maximum feasible generation length L_{gen} and batch size B under the 40GB memory budget.

As shown in Figure 6, PN-Mix supports $L_{\text{gen}} = 32\text{K}$ at $B = 32$ and $B = 110$ at $L_{\text{gen}} = 8\text{K}$, which are $8\times$ longer and $5.5\times$ larger than FP16, respectively. This larger feasible batch range also improves maximum throughput. Appendix E shows that FP16 is faster at the same batch size when it fits in memory, whereas PN-Mix supports larger batches under the 40GB memory budget and reaches $2.37\times$ the maximum per-GPU throughput achievable by FP16.

End-to-end latency. Table 5 reports four representative (B, P, L_{gen}) settings, testing prefill overhead under long prompts and decode-time savings relative to Kitty. For $P = 16\text{K}$, PN-Mix differs from Kitty in TTFT by at most 12 ms, suggesting negligible prefill-stage prediction overhead. Figure 7 shows that PN-Mix reduces marginal per-step latency growth with KV-cache length by 8.8% over Kitty on the long-CoT trajectory ($B = 1$, $P = 141$, $L_{\text{gen}} = 32\text{K}$). This translates into 4.0–6.5% lower decode time across the three $B = 1$ settings. At $B = 32$, $P = 141$, $L_{\text{gen}} = 32\text{K}$, only PN-Mix fits in memory, using 37.12 GB.

[2] https://github.com/Summer-Summer/Kitty

$B/P/L_{\text{gen}}$	Method	Mem (GB)	TTFT (ms)	Decode (s)	Total (s)
1/141/32K	FP16	19.80	46	1299	1299
	Kitty	16.03	55	2121	2121
	PN-Mix	15.95	71	2036	2036
1/16K/8K	FP16	20.26	1679	325	327
	Kitty	17.45	1702	590	592
	PN-Mix	17.39	1714	563	565
1/16K/32K	FP16	23.63	1676	1302	1304
	Kitty	17.99	1706	3254	3256
	PN-Mix	17.87	1715	3042	3044
32/141/32K	FP16	*OOM*	–	–	–
	Kitty	*OOM*	–	–	–
	PN-Mix	37.12	396	3943	3943

Table 5: **End-to-end latency and peak memory on a single A100-SXM4-40GB.** B, P, and L_{gen} denote batch size, prompt length, and generation length. P=141 corresponds to a single AIME24 problem, and P=16K to a GSM8K 88-shot prompt.

7 Conclusion

We presented PN-Mix, a pair-norm-based mixed-precision key-cache quantization method for RoPE-based decoder-only LLMs, designed to support efficient long-output inference. PN-Mix allocates higher precision to RoPE channel pairs with large query–key pair-norm products and estimates these salient pairs once during prefill, avoiding offline calibration and repeated decoding-time saliency recomputation. Across standard reasoning, code generation, and AIME benchmarks, PN-Mix achieves comparable or better average accuracy than prior mixed-precision baselines with lower average key-cache precision. It also remains robust in the sub-2.25-bit regime and expands the feasible serving region for long-output inference.

Limitations

PN-Mix is most directly applicable to RoPE-based decoder-only LLMs, as its saliency score is defined over RoPE channel pairs. Extending the same formulation to models without RoPE-style channel pairs may require additional adaptation.

Our efficiency results are measured on a single NVIDIA A100-SXM4-40GB GPU with our current

Triton-based implementation. Since wall-clock latency depends on hardware, kernel optimization, batching strategy, and serving infrastructure, the exact speedup of PN-Mix may vary across deployment environments.

Acknowledgements

This paper was a result of the research project supported by SK hynix Inc.

References

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

Zefan Cai, Wen Xiao, Hanshi Sun, Cheng Luo, Yikai Zhang, Ke Wan, Yucheng Li, Yeyang Zhou, Li-Wen Chang, Jiuxiang Gu, Zhen Dong, Animashree Anandkumar, Abedelkadir Asi, and Junjie Hu. 2025. R-kv: Redundancy-aware kv cache compression for reasoning models. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 60980–61005. Curran Associates, Inc.

Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Chong-Yan Chen, Yu-Fang Hu, Pei-Shuo Wang, Ning-Chi Huang, Luis Ceze, Mohamed S. Abdelfattah, and Kai-Chiang Wu. 2025. Palu: KV-cache compression with low-rank projection. In *The Thirteenth International Conference on Learning Representations*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *CoRR*, abs/2107.03374.

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, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: fast and memory-efficient exact attention with io-awareness. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

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, and 5 others. 2023. A framework for few-shot language model evaluation.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Mohsen Hariri, Alan Luo, Weicong Chen, Tianyi Zhang, Qifan Wang, Xiaotian Han, and Vipin Chaudhary. 2026. Quantize what counts: More for keys, less for values. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 26393–26420, San Diego, California, United States. Association for Computational Linguistics.

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 *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

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 *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA. Curran Associates Inc.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. 2024. RULER: What's the real context size of your long-context language models? *CoRR*, abs/2404.06654.

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

Shibo Jie, Yehui Tang, Kai Han, Zhi-Hong Deng, and Jing Han. 2025. SpeCache: speculative key-value caching for efficient generation of llms. In *Proceedings of the 42nd International Conference on Machine Learning*, ICML'25. JMLR.org.

Hao Kang, Qingru Zhang, Souvik Kundu, Geonhwa Jeong, Zaoxing Liu, Tushar Krishna, and Tuo Zhao. 2024. GEAR: An efficient error reduction framework for KV cache compression in LLM inference. In *Proceedings of The 4th NeurIPS Efficient Natural Language and Speech Processing Workshop*, volume 262 of *Proceedings of Machine Learning Research*, pages 305–321. PMLR.

Jaeseong Lee, Seung-won Hwang, Aurick Qiao, Daniel Campos, Zhewei Yao, and Yuxiong He. 2025. TALE: Token-adaptive low-rank kvcache approximation with reconstruction elimination. *Transactions of the Association for Computational Linguistics*, 13:1298–1318.

Xing Li, Zeyu XING, Yiming Li, Linping Qu, Hui-Ling Zhen, Yiwu Yao, Wulong Liu, 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 *Forty-second International Conference on Machine Learning*.

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. In *Advances in Neural Information Processing Systems*, volume 37, pages 22947–22970.

Chengxi Liao and Zeyi Wen. 2026. Channel-aware mixed-precision quantization for efficient long-context inference. In *The Fourteenth International Conference on Learning Representations*.

Ruikang Liu, Yuxuan Sun, Manyi Zhang, Haoli Bai, Xianzhi Yu, Tiezheng YU, Chun Yuan, and Lu Hou. 2025. Quantization hurts reasoning? an empirical study on quantized reasoning models. In *Second Conference on Language Modeling*.

Tengxuan Liu, Shiyao Li, Jiayi Yang, Tianchen Zhao, Feng Zhou, Xiaohui Song, Guohao Dai, Shengen Yan, Huazhong Yang, and Yu Wang. 2026. PM-KVQ: Progressive mixed-precision KV cache quantization for long-cot LLMs. In *The Fourteenth International Conference on Learning Representations*.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen (Henry) 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'24. JMLR.org.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, and 24 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

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

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. 2023. Language models are multilingual chain-of-thought reasoners. In *The Eleventh International Conference on Learning Representations*.

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

Yi Su, Yuechi Zhou, Quantong Qiu, Juntao Li, Qingrong Xia, Ping Li, Xinyu Duan, Zhefeng Wang, and Min Zhang. 2025a. Accurate kv cache quantization with outlier tokens tracing. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12895–12915.

Zunhai Su, Hanyu Wei, Zhe Chen, Wang Shen, Linge Li, Huangqi Yu, and Kehong Yuan. 2025b. RotateKV: accurate and robust 2-bit kv cache quantization for llms via outlier-aware adaptive rotations. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, IJCAI '25.

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

Zheng Wang, Boxiao Jin, Zhongzhi Yu, and Minjia Zhang. 2024. Model tells you where to merge: Adaptive kv cache merging for llms on long-context tasks. *arXiv preprint arXiv:2407.08454*.

Songhao Wu, Ang Lv, Xiao Feng, Yufei Zhang, Xun Zhang, Guojun Yin, Wei Lin, and Rui Yan. 2025. PolarQuant: Leveraging polar transformation for key cache quantization and decoding acceleration. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Haojun Xia, Xiaoxia Wu, Jisen Li, Tsai-chuan Wu, Junxiong Wang, Jue Wang, Chenxi Li, Aman Singhal, Alay Dilipbhai Shah, Alpay Ariyak, Donglin Zhuang, Zhongzhu Zhou, Ben Athiwaratkun, Zhen Zheng, and Shuaiwen Leon Song. 2026. Kitty: Accurate and efficient 2-bit kv cache quantization with dynamic channel-wise precision boost. In *Proceedings of Machine Learning and Systems*, volume 8, pages 511–525. MLSys.

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

Yuhui Xu, Zhanming Jie, Hanze Dong, Lei Wang, Xudong Lu, Aojun Zhou, Amrita Saha, Caiming Xiong, and Doyen Sahoo. 2025. ThinK: Thinner key cache by query-driven pruning. In *The Thirteenth International Conference on Learning Representations*.

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.

June Yong Yang, Byeongwook Kim, Jeongin Bae, Beomseok Kwon, Gunho Park, Eunho Yang, Se Jung Kwon, and Dongsoo Lee. 2024. No token left behind: Reliable kv cache compression via importance-aware mixed precision quantization. *Preprint*, arXiv:2402.18096.

Amir Zandieh, Majid Daliri, and Insu Han. 2025. QJL: 1-bit quantized JL transform for kv cache quantization with zero overhead. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(24):25805–25813.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, and Maosong Sun. 2024a. ∞Bench: Extending long context evaluation beyond 100K tokens. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15262–15277, Bangkok, Thailand. Association for Computational Linguistics.

Yifan Zhang and Team Math-AI. 2024. American invitational mathematics examination (aime) 2024. https://huggingface.co/datasets/math-ai/aime24.

Yifan Zhang and Team Math-AI. 2025. American invitational mathematics examination (aime) 2025. https://huggingface.co/datasets/math-ai/aime25.

Yike Zhang, Zhiyuan He, Huiqiang Jiang, Chengruidong Zhang, Yuqing Yang, Jianyong Wang, and Lili Qiu. 2025. LeanK: Learnable k cache channel pruning for efficient decoding. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 31122–31137, Suzhou, China. Association for Computational Linguistics.

Yuxin Zhang, Yuxuan Du, Gen Luo, Yunshan Zhong, Zhenyu Zhang, Shiwei Liu, and Rongrong Ji. 2024b. CaM: Cache merging for memory-efficient LLMs inference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 58840–58850. PMLR.

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. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 34661–34710.

A Analysis on Key- and Value-Cache Quantization Sensitivity

Table 8 compares key-only and value-only KV cache quantization under the same average bit-width using KIVI (Liu et al., 2024). Across both models and AIME benchmarks, value-only quantization is less harmful than key-only quantization, especially when the first 32 tokens are preserved in FP16. These results indicate that low-bit key-cache quantization is the main source of accuracy degradation, supporting our focus on mixed-precision key-cache quantization.

B Key-Cache Quantization Simulation Details

To set up the simulation, we first collect representative inference traces from each model. For R1-Qwen-7B and R1-Llama-8B, we randomly select one problem from AIME25 and LiveCodeBench v5, respectively. We run each model on the selected problem and collect the post-RoPE query and key states from all layers and heads throughout inference.

Table 9 provides the full results of the key-cache quantization simulation used in §4.2 and §4.3. We report key-cache MSE $\Delta_{\tilde{K}}$ and attention KL divergence D_{KL}^{A} for uniform-precision baselines and H3(50%)L1 mixed-precision settings. Uniform baselines include per-channel asymmetric quantization following KIVI (Liu et al., 2024) and per-channel quantization after polar-coordinate conversion following PolarQuant (Wu et al., 2025). For mixed precision, all settings use the same bit-width and differ only in the importance score and whether importance is computed from the full sequence or prefill tokens only.

The results show that $\tilde{Q}\tilde{K}$ pair norm consistently yields the lowest attention KL divergence among the mixed-precision criteria. Importantly, the prefill-token-based version closely matches the full-sequence version, supporting our choice of estimating salient channel pairs once during prefill.

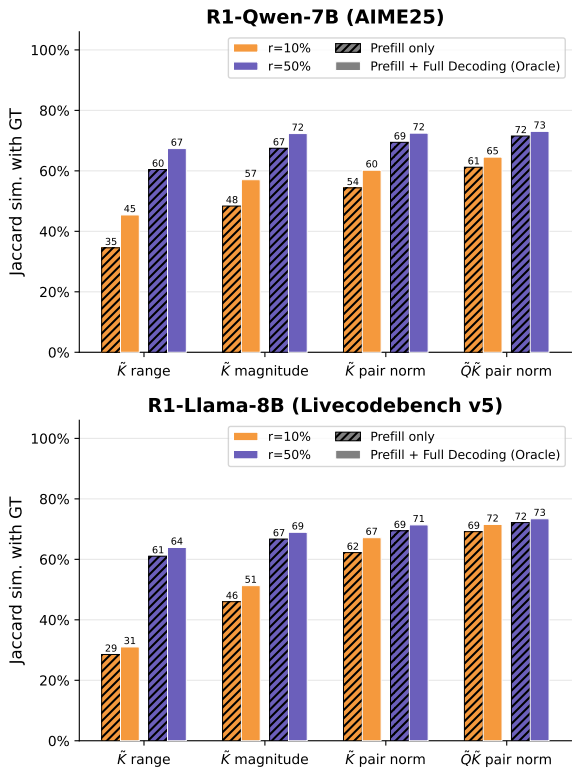

Figure 4: **Salient channel-pair prediction accuracy.** Jaccard similarity between the top-$r\%$ channel-pair predictions of four importance criteria and the decoding-time ground truth, averaged over layers, heads, and decoding steps. Results are shown for $r = 10\%$ and $r = 50\%$.

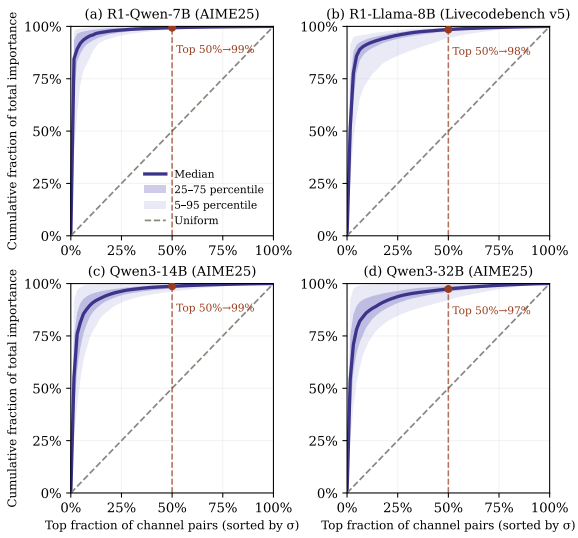

Figure 5: **Cumulative ratio of channel-pair importance scores $\sigma_{h,i}$.** The figure reports the cumulative importance ratio captured by the top $r\%$ channel pairs for R1-Qwen-7B, Qwen3-14B, and Qwen3-32B on AIME25, and for R1-Llama-8B on LiveCodeBench v5. For each layer and KV head, we compute the cumulative importance ratio from prefill-based scores and report the median of these ratios over layers, KV heads, and prompts.

C Salient Channel-Pair Prediction Accuracy

We assess how accurately different importance criteria identify salient RoPE channel pairs during decoding. For each decoding-time query–key pair (t, s), the ground-truth salient set is defined as the top-$r\%$ channel pairs ranked by $\|\mathbf{q}_i^t\|_2 \|\mathbf{k}_i^s\|_2$. Prediction accuracy is quantified by the Jaccard similarity between this set and the top-$r\%$ predictions obtained using four importance criteria: $\tilde{K}$ range, $\tilde{K}$ magnitude, $\tilde{K}$ pair norm, and $\tilde{Q}\tilde{K}$ pair norm. For each criterion, two prediction settings are considered. The first uses only prefill states to make the prediction, while the second uses the entire sequence, including prefill and all decoding states. The Jaccard similarity of both predictions is computed against the decoding-time salient sets.

Figure 4 shows the results on DeepSeek-R1-Distill-Qwen-7B and DeepSeek-R1-Distill-Llama-8B for $r \in \{10\%, 50\%\}$. Across both models and both values of r, the $\tilde{Q}\tilde{K}$ pair norm achieves the highest Jaccard similarity with the decoding-time ground truth, followed by $\tilde{K}$ pair norm, $\tilde{K}$ magnitude, and $\tilde{K}$ range. The advantage of the $\tilde{Q}\tilde{K}$ pair norm is larger at $r = 10\%$, indicating that incorporating query-side information is particularly useful for identifying the most salient channel pairs. Moreover, for the $\tilde{Q}\tilde{K}$ pair norm, using only prefill states yields nearly the same Jaccard similarity as using the entire sequence. This suggests that incorporating decoding states provides little additional benefit for salient-pair prediction.

D Concentration of Pair-Norm Importance Scores

PN-Mix assigns higher precision to a subset of salient RoPE channel pairs identified by $\sigma_{h,i}$. Figure 5 therefore reports the cumulative fraction of the total importance captured by the top-$r\%$ channel pairs. For each layer and KV head, we compute this cumulative fraction using importance scores obtained from prefill tokens and report the median across layers, KV heads, and prompts.

Across R1-Qwen-7B and R1-Llama-8B, the importance scores are highly concentrated in a relatively small subset of channel pairs. We further examine this concentration at larger model scales using Qwen3-14B and Qwen3-32B. On Qwen3-

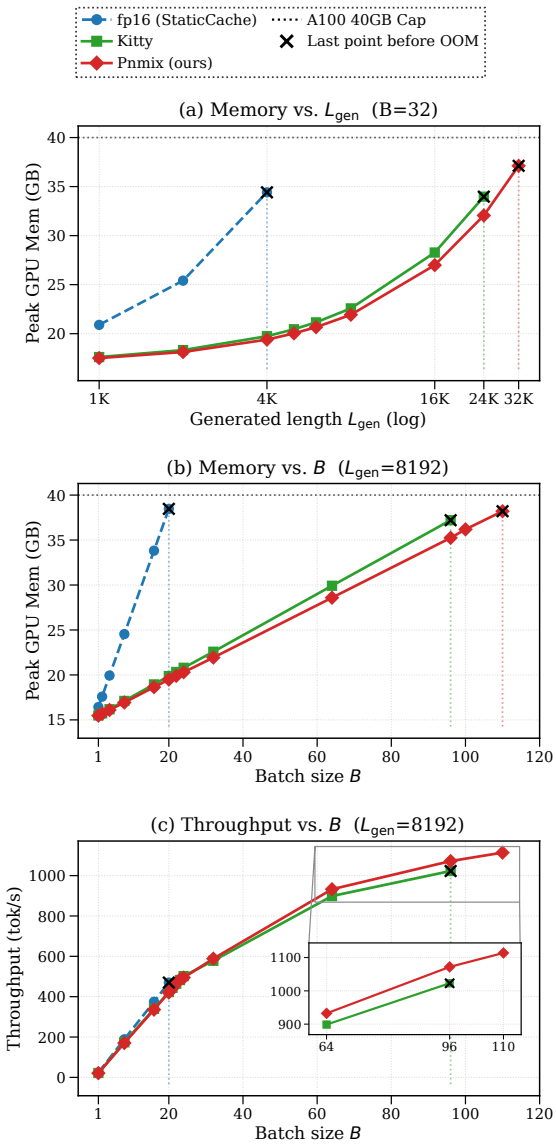

Figure 6: **Memory scaling and generation throughput on a single A100-SXM4-40GB GPU.** We compare FP16, Kitty, and PN-Mix on Qwen3-8B by sweeping (a) generation length L_{gen} at fixed batch size $B = 32$, (b) batch size B at fixed generation length $L_{\text{gen}} = 8\text{K}$, and (c) token generation throughput over B at $L_{\text{gen}} = 8\text{K}$.

14B, the top 10%, 30%, and 50% channel pairs account for 89.3%, 97.0%, and 98.7% of the total importance, respectively. On Qwen3-32B, the corresponding ratios are 86.0%, 94.8%, and 97.4%, showing that the importance distribution remains highly concentrated at 32B scale.

E Memory Scaling and Throughput Analysis

Figure 6 evaluates how peak GPU memory and generation throughput scale with generation length and batch size on Qwen3-8B using a single NVIDIA A100-SXM4-40GB GPU. We compare FP16, Kitty, and PN-Mix. For all methods, the KV cache is pre-allocated up to `max_cache_len` before inference, allowing peak memory to be compared under the same static cache-allocation protocol.

Panel (a) shows peak GPU memory as the generation length L_{gen} increases at a fixed batch size of $B = 32$. The memory usage of FP16 grows rapidly with generation length and exceeds the 40GB GPU memory budget beyond $L_{\text{gen}} = 4\text{K}$. Both Kitty and PN-Mix reduce this growth through KV-cache quantization, with PN-Mix showing the lowest memory usage among the three methods. PN-Mix remains within the memory budget up to $L_{\text{gen}} = 32\text{K}$, enabling an $8\times$ longer generation than FP16 at the same batch size.

Panel (b) considers the complementary setting, fixing $L_{\text{gen}} = 8\text{K}$ while increasing the batch size. Both quantized methods support substantially larger batches than FP16, with PN-Mix maintaining the lowest memory usage as B increases. FP16 supports at most $B = 20$ within the 40GB memory budget, whereas PN-Mix scales to $B = 110$. Thus, PN-Mix supports a $5.5\times$ larger batch size than FP16 at the same generation length. Together, panels (a) and (b) show that reducing the KV-cache footprint expands the feasible operating region along both the generation-length and batch-size dimensions.

Lower memory usage, however, does not necessarily imply higher throughput at the same batch size. Panel (c) therefore reports per-GPU generation throughput as the batch size is increased at $L_{\text{gen}} = 8\text{K}$, using a prompt length of $P = 141$ and greedy decoding.

When the FP16 KV cache fits in memory, FP16 remains faster at the same batch size. For example, at $B = 20$, FP16 achieves 469.8 tok/s compared with 420.8 tok/s for PN-Mix. However, FP16 already consumes 38.48 GB at this point and runs out of memory at $B = 21$, limiting its maximum feasible throughput to 469.8 tok/s.

PN-Mix can continue increasing the batch size because of its smaller KV-cache footprint. It exceeds the maximum feasible FP16 throughput at $B = 23$, reaching 485.0 tok/s, and scales to $B = 110$, where it achieves 1114.0 tok/s while using 38.21 GB of GPU memory. This corresponds to $2.37\times$ the maximum per-GPU throughput achievable by FP16 under the same 40GB memory budget.

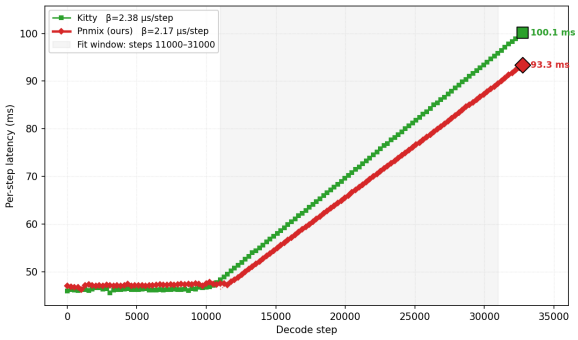

Figure 7: **Per-step decode latency on a single A100-SXM4-40GB GPU.** We measure Qwen3-8B decode latency with Kitty and PN-Mix. The shaded region (steps 11K–31K) indicates the KV-cache-read-dominated regime, and β denotes the linear-regression slope in this region (μs/step).

These results distinguish two deployment regimes. When GPU memory is sufficient for the desired batch size and generation length, FP16 provides higher same-batch throughput because it avoids quantization and dequantization overhead. In memory-bound settings with larger batches or longer generations, PN-Mix can run configurations that exceed the FP16 memory limit. This larger feasible batch size allows PN-Mix to achieve a higher maximum per-GPU throughput under the same memory budget.

F Per-Step Decoding Latency Analysis

Figure 7 compares the per-step decoding latency of PN-Mix and Kitty along a long-CoT trajectory with $B = 1$, $P = 141$, and $L_{\text{gen}} = 32\text{K}$, measured using `torch.cuda.Event`. Both methods remain at a roughly 45–48 ms plateau up to approximately 10K decoding steps. Beyond this point, per-step latency increases steadily with the growing KV-cache length. Over the linear region from 11K to 31K steps, PN-Mix exhibits a lower latency-growth slope than Kitty (2.17 vs. 2.38 μs/step), corresponding to an 8.8% reduction. At 32K decoding steps, PN-Mix is also 6.8 ms faster than Kitty. These results show that PN-Mix's lower average key-cache bit-width and absence of page-wise salient-channel index I/O lead to practical decoding latency improvements.

G Kernel Implementation

We implement all kernels in Triton, including quantization and bit packing of the key and value caches, as well as the fused dequantization and attention operations during decoding.

The kernels are implemented for the configuration used in our downstream task evaluation, namely $b_H = 3$, $b_L = 1$, and $b_V = 2$. After mixed-precision quantization of the key cache, salient channel pairs are represented by integers in the range $[0, 7]$, while the remaining channels are represented by integers in the range $[0, 1]$. Since the least significant 1 bit is shared by all channels and the upper 2 bits are meaningful only for salient channel pairs, our implementation packs these two parts into separate `uint8` tensors. Specifically, we first extract the LSBs of all channels and pack every 8 consecutive token values of each channel into one byte, producing `packed_low`, which is stored for all channels. We then extract the upper 2 bits and pack every 4 consecutive token values of each channel into one byte, producing `packed_hi`, which is stored only for the salient channel pairs. For the value cache, after quantization, every 4 consecutive channel values of each token are packed into one byte and stored as a `uint8` tensor.

During the decoding-stage attention computation, we launch two Triton kernels: one for computing $\tilde{Q}_{h,t}\tilde{K}_{h,1:t}^{\top}/\sqrt{d}$ and the other for computing $A_{h,t}V_{h,1:t}$. Each kernel sequentially processes the sink-token region, the quantized-cache region, and the residual-buffer region. For the quantized region, the dequantized key or value cache is never written back to global memory. Instead, each quantization group is loaded into on-chip registers, dequantized, and immediately consumed by the dot-product computation.

H Average Key Bit-Width Calculation and Method Configurations

To compare the methods under a similar key-cache memory budget, we include both the quantized key values and the additional metadata required by each method in the bit-width calculation. Since the methods use different native precision-allocation schemes, Tables 2 and 3 retain these schemes and use configurations within a narrow 2.25–2.4-bit range rather than enforcing an exact bit match. The resulting comparison therefore focuses on accuracy preservation in this low-bit regime rather than on the best-performing or recommended operating point of each method.

For a consistent calculation across methods, we exclude the FP16 residual buffer from both the storage cost and the number of key elements used

for normalization. Let T denote the total number of tokens, N the FP16 residual-buffer length, N_s the number of sink tokens retained in FP16, and

$$T_q = T - N_s - N \tag{8}$$

the number of quantized tokens. We further denote the token-axis quantization group size by G and the head dimension by d. The effective key-storage cost includes the quantized key payload, FP16 sink tokens when applicable, and scale/zero-point storage. Saliency-related indices or masks are included separately as metadata overhead, and the sum of the two is reported as the total average key-cache bit-width. For sequence-length-dependent terms, we use $T_q = 32{,}768$.

Uniform-precision methods. KIVI and PolarQuant use a uniform key bit-width b and do not require saliency metadata. For each quantized key element, the payload contributes b bits. Since each channel stores one FP16 scale and one FP16 zero-point for every group of G tokens, the scale/zero-point overhead is $32/G$ bits per key element. Their effective key-storage cost is therefore

$$\bar{b}_{K,\text{storage}}^{\text{KIVI/PolarQuant}} = b + \frac{32}{G}. \tag{9}$$

Kitty and PN-Mix. Both Kitty and PN-Mix preserve N_s sink tokens in FP16 and use mixed-precision key quantization with high and low precisions b_H and b_L. For both methods, the effective key-storage cost consists of three components: the FP16 sink tokens, the mixed-precision quantized key values, and the FP16 scale/zero-point values stored for each token group. We define the average storage cost per quantized key element as

$$b_{\text{mix}} = r_{\text{eff}} b_H + (1 - r_{\text{eff}}) b_L + \frac{32}{G}. \tag{10}$$

Here, the first two terms represent the mixed-precision key payload, while $32/G$ accounts for the FP16 scale and zero-point stored for every group of G tokens. Normalizing the total storage by the number of key elements excluding the residual buffer gives

$$\bar{b}_{K,\text{storage}} = \frac{16N_s + T_q b_{\text{mix}}}{N_s + T_q}. \tag{11}$$

The two methods differ in how the high-precision channels are selected and how the corresponding saliency information is stored. For Kitty, high precision is assigned at the channel level. Because the number of selected channels must be an integer, the effective high-precision ratio is

$$r_{\text{eff}}^{\text{Kitty}} = \frac{\lfloor dr \rfloor}{d}, \tag{12}$$

where r is the nominal selection ratio. Kitty stores a channel-selection index for every quantization page. Since each page stores one `uint8` index per channel, the metadata cost is distributed over the G tokens in the page, resulting in

$$\bar{b}_{K,\text{meta}}^{\text{Kitty}} = \frac{8T_q}{G(N_s + T_q)}. \tag{13}$$

For PN-Mix, high precision is assigned at the RoPE-pair level, so the selected number of channels must be an even integer. The corresponding effective high-precision ratio is

$$r_{\text{eff}}^{\text{PN-Mix}} = \frac{2\lfloor rd/2 \rfloor}{d}. \tag{14}$$

Unlike Kitty, PN-Mix determines the salient RoPE pairs once during prefill and shares the resulting mask across all quantization groups. Since there are $d/2$ RoPE pairs and each pair requires one bit to record whether it is assigned high precision, the total mask size is $d/2$ bits per KV head. Dividing this one-time mask cost by the number of key elements excluding the residual buffer gives the average metadata overhead

$$\bar{b}_{K,\text{meta}}^{\text{PN-Mix}} = \frac{1}{2(N_s + T_q)}, \tag{15}$$

which becomes negligible for long sequences.

ChanMix. ChanMix starts from a 2-bit allocation for all channels and promotes selected retrieval KV heads to 4 bits. Among the remaining channels, the same number of channels are reassigned to 1 and 3 bits, so their average precision remains 2 bits. If $r_{\text{eff}}^{\text{CM}}$ denotes the actual fraction of promoted KV heads after integer rounding, the average payload is therefore

$$4r_{\text{eff}}^{\text{CM}} + 2\left(1 - r_{\text{eff}}^{\text{CM}}\right) = 2\left(1 + r_{\text{eff}}^{\text{CM}}\right). \tag{16}$$

The effective promoted-head ratio is

$$r_{\text{eff}}^{\text{CM}} = \frac{\lfloor LH^{kv} r_h \rfloor}{LH^{kv}}, \tag{17}$$

where r_h is the nominal retrieval-head ratio, L is the number of layers, and H^{kv} is the number of

KV heads per layer. ChanMix stores its scale and zero-point values in FP8, so each token group contributes 16 bits per channel for these two values, or $16/G$ bits per key element. Its effective key-storage cost is thus

$$\bar{b}_{K,\text{storage}}^{\text{ChanMix}} = 2\left(1 + r_{\text{eff}}^{\text{CM}}\right) + \frac{16}{G}. \tag{18}$$

ChanMix additionally stores, for each channel, its assigned bit-width and an index that maps the quantized channel ordering back to the original channel ordering. These two metadata tensors are stored once and shared across all tokens. Their average metadata overhead is

$$\bar{b}_{K,\text{meta}}^{\text{ChanMix}} = \frac{64}{N_s + T_q}, \tag{19}$$

which is negligible at the sequence lengths considered here.

Table 10 summarizes the exact configurations used in our experiments and the resulting bit-width calculation. Panel (a) reports the method-specific quantization and selection parameters, including the effective selection ratios after integer rounding. Panel (b) separately reports the effective key-storage cost and saliency-metadata overhead, whose sum gives the total average key-cache bit-width. The resulting configurations span approximately 2.25–2.39 average key bits.

For PolarQuant, the 2-bit radius/angle configuration is used specifically to place the method in this low-bit comparison regime. PolarQuant originally uses 3-bit per-channel quantization, and reducing both radius and angle channels to 2 bits can amplify quantization error after the polar-coordinate transformation. We therefore do not interpret the results in Tables 2 and 3 as comparisons against each method's recommended operating point; rather, they evaluate accuracy preservation under comparable low key-cache budgets.

For ChanMix, official profiles are unavailable for Qwen3-8B/32B and DeepSeek-R1-Distill-Qwen-7B. We therefore reproduce the profiling procedure from the original ChanMix method and validate it on three models with official profiles. The reproduced profiles show Pearson correlations of 0.98–0.99 with the official profiles for both key-range and retrieval-head scores, with 90–91% overlap in the selected top-30% retrieval heads. We then use the validated procedure for the three models without official profiles. Results based on these reproduced profiles are marked with † in Tables 2 and 3.

I Configurations for the Bit-Rate Pareto Frontier

Table 14 reports the detailed hyperparameter configurations and AIME24 accuracies for the operating points shown in Figure 3. For each method, we sweep candidate configurations and select the operating point whose average key-cache bit-width is closest to each target bit budget: 3.0, 2.5, 2.25, 2.0, and 1.5 bits. For PN-Mix, we sweep $b_H, b_L \in \{4, 3, 2, 1\}$, $G \in \{64, 128\}$, and $r \in \{0.1, 0.2, \ldots, 0.9\}$. For KIVI, we sweep $b \in \{2, 1\}$ and $G \in \{32, 64, 128\}$.

For Kitty, we follow its original design in the $\geq$2.25-bit regime by fixing $(b_H, b_L) = (4, 2)$ and sweeping only G and r. In the sub-2.25-bit regime, where the original $(4, 2)$ setting cannot reach the target bit budgets, we additionally sweep (b_H, b_L) using the same candidate set as PN-Mix. For both Kitty and PN-Mix, the first $N_s = 32$ sink tokens are preserved in full precision, while the value cache is kept at $b_V = 16$ for all operating points.

J Additional Evaluation on Long-Input and Multilingual Tasks

To evaluate whether PN-Mix generalizes beyond the reasoning-oriented long-output settings in the main experiments, we further evaluate it on long-input and multilingual tasks using Llama-3.1-8B-Instruct. We compare PN-Mix with KIVI, ChanMix, and Kitty under the same low-bit regime: all compressed methods use an average key-cache precision of 2.25–2.39 bits, while the value cache is kept in full precision.

RULER. Table 13 reports results on all 13 RULER (Hsieh et al., 2024) tasks at context lengths from 16K to 64K. PN-Mix achieves the highest average accuracy among the compressed methods at all context lengths and remains close to FP16 even at 64K. The advantage is particularly pronounced on the challenging multi-key retrieval task (MK3), where PN-Mix maintains nearly unchanged accuracy as the context grows, while the competing quantized methods degrade substantially.

InfiniteBench. Table 12 presents results on InfiniteBench (Zhang et al., 2024a), covering diverse long-input tasks including retrieval, dialogue QA, long-document QA, and summarization. PN-Mix (2) achieves the highest average score among the compressed methods (33.98), close to FP16 (34.15), demonstrating that its robustness extends

Method	en	zh	fr	ja	bn
FP16	82.80	62.80	64.00	50.80	58.00
KIVI	73.20	56.00	58.80	47.20	44.40
ChanMix	74.80	54.40	55.20	45.60	50.40
Kitty	**80.40**	57.20	62.40	48.80	**55.20**
PN-Mix (1)	**80.40**	57.20	63.60	48.40	52.40
PN-Mix (2)	80.00	**59.20**	**65.20**	**53.60**	52.00

Table 6: **Accuracy comparison on MGSM with Llama-3.1-8B-Instruct.** We evaluate the full 250-example test set using 8-shot native-language CoT. Performance is measured using flexible-extract accuracy. All compressed configurations (i.e., every method except FP16) are matched to an average key-cache precision of 2.25–2.39 bits, while the value cache is kept in full precision. Bold indicates the best non-FP16 result in each column.

Method	**Avg. KV bits**	**AIME24**	**AIME25**
R1-Llama-8B			
FP16	16	$48.33_{\pm 6.19}$	$31.33_{\pm 8.19}$
PM-KVQ	2.38	$46.33_{\pm 6.04}$	$31.66_{\pm 4.53}$
PN-Mix (1)	2.26	$47.67_{\pm 6.16}$	$31.00_{\pm 4.23}$
PN-Mix (2)	2.28	$44.33_{\pm 5.97}$	$33.00_{\pm 5.04}$
R1-Qwen-7B			
FP16	16	$54.33_{\pm 5.39}$	$39.33_{\pm 7.57}$
PM-KVQ	2.32	$38.33_{\pm 4.53}$	$31.67_{\pm 4.99}$
PN-Mix (1)	2.26	$47.67_{\pm 3.96}$	$35.00_{\pm 5.82}$
PN-Mix (2)	2.28	$45.67_{\pm 5.39}$	$32.67_{\pm 5.54}$

Table 7: **Comparison with PM-KVQ on AIME24 and AIME25.** For PN-Mix, we use the same $b_V = 2$ configurations as in Table 3, while PM-KVQ uses the lowest model-specific mixed-precision operating point within the 2.25–2.4 average-bit range. Each cell reports the mean $\pm$ standard deviation over 10 responses per question.

beyond controlled retrieval tasks to diverse long-context workloads.

MGSM. We assess multilingual performance on MGSM (Shi et al., 2023) across the compared methods (Table 6). PN-Mix remains competitive across all evaluated languages, with PN-Mix (2) achieving the best compressed-model accuracy on Chinese, French, and Japanese, showing that its effectiveness is not limited to English generation.

Taken together, these results indicate that PN-Mix maintains a favorable accuracy–precision trade-off across diverse long-context and multilingual workloads.

K Additional Comparison with PM-KVQ

We include a direct comparison with PM-KVQ (Liu et al., 2026), a layer-level mixed-precision KV-cache quantization method for long-CoT LLMs. PM-KVQ profiles the quantization sensitivity of each transformer block on calibration data and uses integer programming to assign a final bit-width (Fbit) to each block under a given memory budget. During inference, it progressively reduces the KV-cache precision from 16 bits toward the assigned Fbit as the block-level memory budget is filled. In contrast, PN-Mix allocates precision within each attention head at RoPE channel-pair granularity using input-adaptive saliency estimated during prefill, without offline calibration.

Following the comparison protocol used in our main experiments, we evaluate the methods within the 2.25–2.4 average-bit range under the $b_V = 2$ setting. PM-KVQ uses layer-level 4/2-bit Fbit allocation, which admits only discrete operating points. Assigning a 2-bit Fbit to every layer removes cross-layer precision differentiation, so we instead use the lowest configurations that retain PM-KVQ's mixed-precision allocation. Specifically, PM-KVQ assigns a 4-bit Fbit to 2 of 32 layers for R1-Llama-8B and to 1 of 28 layers for R1-Qwen-7B, resulting in average KV bit-widths of approximately 2.38 and 2.32 bits, respectively. These configurations still give PM-KVQ a larger average KV-bit budget than PN-Mix, which operates at 2.26–2.28 bits.

As shown in Table 7, PN-Mix achieves comparable accuracy to PM-KVQ on R1-Llama-8B despite using the smaller average KV-bit budget. On R1-Qwen-7B, both PN-Mix variants achieve higher mean accuracy than PM-KVQ on AIME24 and AIME25 while using fewer average KV bits.

L Step-Wise Stability of Prefill-Based Saliency Selection

Appendix C shows that channel pairs selected from the prefill states exhibit high overlap with decoding-time salient channel pairs on average. However, this averaged overlap does not reveal whether the overlap between the channel pairs selected during prefill and decoding-time salient channel pairs remains stable throughout generation or decreases at later decoding steps. We therefore examine how this overlap changes over the course of decoding under long-generation and multi-topic stress-test settings.

For each causal query–key pair (t, s) at decoding step t, we compare the top-$r\%$ channel pairs selected from the prefill states with the top-$r\%$ salient channel pairs identified for that query–key

pair. To prevent the overlap from being dominated by prefill keys, we restrict s to keys generated during decoding up to step t. We use two reference scores to identify the decoding-time salient channel pairs: the pair-norm score, $\|\mathbf{q}_i^t\|_2\|\mathbf{k}_i^s\|_2$, and the angle-aware absolute pair dot product, $|\langle \mathbf{q}_i^t, \mathbf{k}_i^s \rangle|$. For reporting, we group the decoding steps into four consecutive ranges and average the overlap over all eligible query–key pairs, heads, and layers within each range. We additionally report the difference between the maximum and minimum overlap across 1K-step bins for the long-generation setting and across the four reporting bins for the multi-topic setting to summarize temporal variation.

Table 11 shows that the overlap between the channel pairs selected during prefill and decoding-time salient channel pairs remains stable throughout decoding in both stress-test settings. In the long-generation setting, where R1-Qwen-7B generates 32,672 tokens from only 96 prefill tokens, the overlap for the $\tilde{Q}\tilde{K}$ pair-norm criterion at $r = 50\%$ stays at approximately 0.72 across all four decoding ranges under the pair-norm score and at approximately 0.68 under the absolute pair dot product. This indicates that the overlap does not progressively decrease over the long decoding trajectory. The same pattern holds in the multi-topic setting containing 60 questions from 57 MMLU subjects. The overlap remains approximately 0.75 under the pair-norm score and approximately 0.71 under the absolute pair dot product across all four decoding ranges. The overlap therefore remains stable not only under long generation but also when the prompt spans many different topics. Across both stress-test settings, the $\tilde{Q}\tilde{K}$ pair-norm criterion also maintains the highest overlap among the compared selection criteria. The small differences between the maximum and minimum overlap further confirm limited temporal variation. A similar pattern at $r = 10\%$ shows that this behavior is not specific to the $r = 50\%$ selection ratio.

Overall, these results indicate that the overlap between the channel pairs selected during prefill and decoding-time salient channel pairs remains relatively stable throughout decoding in the tested long-generation and multi-topic settings.

M Use of AI Assistants

AI assistants were used only for language editing, including grammar and spelling checks and synonym suggestions, and for assistance with Triton code implementation. All technical content and implementation details were verified and finalized by the authors.

Model	Task	K16V16	K2.25V16	K2.25*V16	K16V2.25	K16V2.25*
R1-Llama-8B	AIME24	$48.33_{\pm 6.19}$	$28.67_{\pm 7.02}$	$44.00_{\pm 7.27}$	$40.67_{\pm 5.33}$	$\mathbf{51.00}_{\pm \mathbf{6.51}}$
	AIME25	$31.33_{\pm 8.19}$	$26.67_{\pm 5.16}$	$28.67_{\pm 3.71}$	$26.33_{\pm 4.58}$	$\mathbf{32.00}_{\pm \mathbf{5.81}}$
Qwen3-8B	AIME24	$76.33_{\pm 4.82}$	$64.00_{\pm 5.54}$	$70.00_{\pm 2.98}$	$68.33_{\pm 4.53}$	$\mathbf{73.33}_{\pm \mathbf{2.98}}$
	AIME25	$67.00_{\pm 4.58}$	$55.33_{\pm 4.99}$	$59.67_{\pm 4.07}$	$60.00_{\pm 2.11}$	$\mathbf{68.00}_{\pm \mathbf{5.42}}$

Table 8: **Sensitivity of key/value caches to quantization.** Each cell reports mean $\pm$ std over 10 responses per question. All quantized methods use KIVI (Liu et al., 2024) ($G = 128$, $N = 128$); method names ending with (*) preserve the first 32 tokens in full precision. The best quantized method per row within each model block is in **bold**.

Method	R1-Qwen-7B (AIME25)		R1-Llama-8B (LiveCodeBench v5)	
	$\Delta_{\tilde{K}}$	D_{KL}^{A}	$\Delta_{\tilde{K}}$	D_{KL}^{A}
Only 3-bit (Uniform precision)				
Per-Channel	0.042	0.043	0.058	0.020
Per-Channel after (r, θ)	0.098	0.101	0.142	0.043
Only 2-bit (Uniform precision)				
Per-Channel	0.230	0.232	0.324	0.120
Per-Channel after (r, θ)	0.499	0.518	0.722	0.213
H3(50%)L1 (Full token based)				
$\tilde{K}$ range	$0.184_{(0.913)}$	$0.125_{(0.378)}$	$0.286_{(1.453)}$	$0.094_{(0.330)}$
$\tilde{K}$ magnitude	$0.194_{(0.975)}$	$0.087_{(0.294)}$	$0.307_{(1.588)}$	$0.065_{(0.294)}$
$\tilde{K}$ pair norm	$0.194_{(0.976)}$	$0.083_{(0.291)}$	$0.313_{(1.629)}$	$0.060_{(0.293)}$
$\tilde{Q}\tilde{K}$ pair norm	$0.197_{(0.998)}$	$0.080_{(0.273)}$	$0.320_{(1.668)}$	$0.055_{(0.279)}$
H3(50%)L1 (Prefill token based)				
$\tilde{K}$ range	$0.193_{(0.966)}$	$0.128_{(0.473)}$	$0.281_{(1.430)}$	$0.078_{(0.307)}$
$\tilde{K}$ magnitude	$0.198_{(0.998)}$	$0.096_{(0.362)}$	$0.303_{(1.561)}$	$0.062_{(0.299)}$
$\tilde{K}$ pair norm	$0.196_{(0.988)}$	$0.090_{(0.334)}$	$0.309_{(1.602)}$	$0.059_{(0.293)}$
$\tilde{Q}\tilde{K}$ pair norm	$0.201_{(1.022)}$	$0.082_{(0.283)}$	$0.317_{(1.649)}$	$0.055_{(0.278)}$

Table 9: **Key-cache quantization simulation.** Table shows $\Delta_{\tilde{K}}$ (key MSE) and D_{KL}^{A} (attention KL divergence) for uniform-precision and H3(50%)L1 mixed-precision methods on R1-Qwen-7B (AIME25) and R1-Llama-8B (LiveCodeBench v5). All methods use per-channel asymmetric quantization ($G = 128$); 1-bit groups use the correction of Eq. 7. Each H3(50%)L1 entry reports the value *with* 1-bit correction, with the value *without* correction shown as a parenthesized subscript. Mixed-precision rows differ only in the channel(-pair) importance score used to allocate the high/low bits, with importance computed over either the full sequence or prefill tokens only.

(a) Method configurations

Method	b	b_H	b_L	r / r_h	r_{eff}	G	N	N_s
KIVI	2	–	–	–	–	128	128	0
PolarQuant	2	–	–	–	–	128	128	0
ChanMix	–	4	3, 2, 1	0.1	–	128	128	0
Kitty	–	4	2	0.03125	$\frac{\lfloor dr \rfloor}{d}$	128	128	32
PN-Mix (1)	–	3	1	0.5	$\frac{2\lfloor rd/2 \rfloor}{d}$	128	128	32
PN-Mix (2)	–	3	1	0.4	$\frac{2\lfloor rd/2 \rfloor}{d}$	64	128	32

(b) Average key-cache bit-width accounting

Method	**Effective K-storage formula**	**Effective K storage bits**	**Saliency metadata formula**	**Avg. metadata overhead**	**Total K avg. bit**
KIVI	$b + \frac{32}{G}$	2.2500	–	0	2.2500
PolarQuant	$b + \frac{32}{G}$	2.2500	–	0	2.2500
ChanMix	$2\left(1 + \frac{\lfloor LH^{kv} r_h \rfloor}{LH^{kv}}\right) + \frac{16}{G}$	2.3203	$\frac{64}{N_s+T_q}$	≈ 0	2.3203
Kitty	$\frac{16N_s+T_q\left[r_{\text{eff}} b_H+(1-r_{\text{eff}})b_L+\frac{32}{G}\right]}{N_s+T_q}$	2.3259	$\frac{8T_q}{G(N_s+T_q)}$	0.0624	2.3883
PN-Mix (1)	$\frac{16N_s+T_q\left[r_{\text{eff}} b_H+(1-r_{\text{eff}})b_L+\frac{32}{G}\right]}{N_s+T_q}$	2.2634	$\frac{1}{2(N_s+T_q)}$	≈ 0	2.2634
PN-Mix (2)	$\frac{16N_s+T_q\left[r_{\text{eff}} b_H+(1-r_{\text{eff}})b_L+\frac{32}{G}\right]}{N_s+T_q}$	2.2946	$\frac{1}{2(N_s+T_q)}$	≈ 0	2.2946

Table 10: **Method configurations and average key-cache bit-widths.** Average bit-widths are computed with $T_q = 32{,}768$, where T_q denotes the number of quantized tokens; the FP16 residual buffer is excluded from the calculation. For Kitty and PN-Mix, r_{eff} denotes the actual fraction of channels assigned to high precision after converting the nominal ratio r to an integer number of selected channels or RoPE channel pairs, respectively. Effective K-storage bits exclude saliency metadata, which is reported separately and added to obtain the total average K bit-width.

(a) Long generation: R1-Qwen-7B / AIME25, 96 prefill tokens + 32,672 decoding steps						
Reference score	**Selection criterion**	**0–8K**	**8–16K**	**16–24K**	**24K–32.7K**	**Max–Min**
$\|\mathbf{q}_i^t\|_2\|\mathbf{k}_i^s\|_2$	$\tilde{K}$ range	$0.61/0.35$	$0.62/0.35$	$0.61/0.35$	$0.61/0.35$	$0.009/0.029$
	$\tilde{K}$ magnitude	$0.68/0.49$	$0.69/0.50$	$0.69/0.50$	$0.69/0.51$	$0.008/0.034$
	$\tilde{K}$ pair norm	$0.70/0.54$	$0.70/0.56$	$0.70/0.56$	$0.70/0.56$	$0.010/0.040$
	$\tilde{Q}\tilde{K}$ **pair norm**	$\mathbf{0.72/0.62}$	$\mathbf{0.72/0.61}$	$\mathbf{0.72/0.61}$	$\mathbf{0.72/0.61}$	$\mathbf{0.007/0.027}$
$\|\langle\mathbf{q}_i^t, \mathbf{k}_i^s\rangle\|$	$\tilde{K}$ range	$0.59/0.33$	$0.60/0.33$	$0.60/0.33$	$0.60/0.33$	$0.008/0.024$
	$\tilde{K}$ magnitude	$0.65/0.45$	$0.65/0.46$	$0.65/0.46$	$0.65/0.46$	$0.007/0.023$
	$\tilde{K}$ pair norm	$0.66/0.50$	$0.67/0.51$	$0.67/0.50$	$0.67/0.51$	$0.008/0.026$
	$\tilde{Q}\tilde{K}$ **pair norm**	$\mathbf{0.68/0.56}$	$\mathbf{0.68/0.55}$	$\mathbf{0.68/0.54}$	$\mathbf{0.68/0.54}$	$\mathbf{0.006/0.029}$

(b) Multi-topic generation: Llama-3.1-8B-Instruct / LongGenBench-MMLU, 10,366 prefill tokens + 3,970 decoding steps						
Reference score	**Selection criterion**	**0–992**	**992–1984**	**1984–2976**	**2976–3970**	**Max–Min**
$\|\mathbf{q}_i^t\|_2\|\mathbf{k}_i^s\|_2$	$\tilde{K}$ range	$0.65/0.30$	$0.66/0.30$	$0.65/0.29$	$0.65/0.29$	$0.001/0.007$
	$\tilde{K}$ magnitude	$0.71/0.55$	$0.71/0.55$	$0.71/0.56$	$0.71/0.56$	$0.002/0.007$
	$\tilde{K}$ pair norm	$0.73/0.68$	$0.73/0.68$	$0.73/0.68$	$0.73/0.68$	$0.002/0.008$
	$\tilde{Q}\tilde{K}$ **pair norm**	$\mathbf{0.75/0.72}$	$\mathbf{0.75/0.72}$	$\mathbf{0.75/0.72}$	$\mathbf{0.75/0.73}$	$\mathbf{0.002/0.009}$
$\|\langle\mathbf{q}_i^t, \mathbf{k}_i^s\rangle\|$	$\tilde{K}$ range	$0.63/0.27$	$0.63/0.27$	$0.63/0.28$	$0.63/0.28$	$0.002/0.004$
	$\tilde{K}$ magnitude	$0.67/0.46$	$0.68/0.47$	$0.68/0.47$	$0.68/0.48$	$0.003/0.020$
	$\tilde{K}$ pair norm	$0.69/0.56$	$0.69/0.56$	$0.69/0.57$	$0.69/0.58$	$0.003/0.023$
	$\tilde{Q}\tilde{K}$ **pair norm**	$\mathbf{0.71/0.58}$	$\mathbf{0.71/0.59}$	$\mathbf{0.71/0.60}$	$\mathbf{0.71/0.61}$	$\mathbf{0.003/0.024}$

Table 11: **Stability of prefill-selected channel pairs during decoding.** We measure the overlap between the fixed top-$r\%$ channel pairs selected from prefill and the top-$r\%$ channel pairs identified by a step-wise reference during decoding. The reference scores are $\|\mathbf{q}_i^t\|_2\|\mathbf{k}_i^s\|_2$ and $|\langle\mathbf{q}_i^t, \mathbf{k}_i^s\rangle|$. For each decode query at step t, the reference considers only decode-generated causal keys ($P \leq s \leq t$), where P is the prefill length. Each cell reports the overlap ratio aggregated over all eligible query–key interactions, heads, and layers within the bin, in the order $r = 50\%/r = 10\%$. Overlap ranges from 0 to 1, where 1 indicates identical selections. Max–Min is computed over 1K-step bins for the long-generation setting and 992-step bins for the multi-topic setting.

Method	K bits	KV-Retr. (acc)	LongDial.-QA (acc)	LongBook-Choice (acc)	LongBook-QA (F1)	LongBook-Sum (ROUGE-L-Sum)	Avg.
FP16	16	59.00	11.00	62.00	9.66	29.07	34.15
KIVI	2.25	32.00	9.00	58.00	**10.03**	25.55	26.92
ChanMix	2.32	53.00	9.00	58.00	10.01	26.74	31.35
Kitty	2.39	49.00	**10.00**	61.00	8.82	**29.09**	31.58
PN-Mix (1)	2.26	**59.00**	7.00	62.00	8.67	28.76	33.09
PN-Mix (2)	2.29	**59.00**	**10.00**	**63.00**	9.14	28.77	**33.98**

Table 12: **Accuracy comparison on InfiniteBench with Llama-3.1-8B-Instruct.** We evaluate the first 100 examples of each of five English tasks. Prompts are built with the official task templates and middle-truncated to 32,000 tokens using the official truncation routine; no chat template is applied, and decoding is greedy with the official per-task generation budgets (6–1,200 tokens). All compressed configurations (i.e., every method except FP16) are matched to an average key-cache precision of 2.25–2.39 bits, while the value cache is kept in full precision. Bold indicates the best non-FP16 result in each column.

Len.	Method	S1	S2	S3	MK1	MK2	MK3	MV	MQ	FWE	CWE	QA1	QA2	VT	Avg.
16K	FP16	100.0	100.0	100.0	99.5	100.0	100.0	100.0	99.4	95.2	86.6	70.8	57.5	99.8	92.98
	KIVI	100.0	99.5	93.5	99.5	96.5	69.0	95.2	95.4	93.0	84.7	71.5	54.5	95.2	88.26
	ChanMix	100.0	99.5	99.0	99.0	99.0	97.5	97.2	97.5	92.7	87.7	70.8	56.5	98.7	91.93
	Kitty	100.0	100.0	99.5	99.5	98.5	95.0	99.4	98.5	94.3	87.1	72.8	54.5	98.1	92.09
	PN-Mix (1)	100.0	100.0	100.0	100.0	100.0	99.0	100.0	99.4	95.7	85.0	72.6	56.5	99.4	**92.89**
	PN-Mix (2)	100.0	100.0	100.0	99.5	99.5	100.0	99.9	99.1	95.0	82.2	72.1	56.5	99.4	92.55
32K	FP16	100.0	100.0	100.0	100.0	99.5	99.0	99.4	98.8	93.2	22.8	74.6	53.0	98.9	87.62
	KIVI	100.0	100.0	95.0	99.5	95.0	47.5	94.9	92.4	91.3	28.6	69.4	49.5	94.2	81.32
	ChanMix	99.5	100.0	99.0	100.0	98.5	91.5	96.8	98.5	92.7	21.7	71.9	52.0	96.7	86.06
	Kitty	100.0	100.0	98.5	100.0	97.0	82.0	96.6	98.8	92.5	34.5	71.8	51.0	97.6	86.17
	PN-Mix (1)	100.0	100.0	100.0	100.0	99.0	98.5	97.8	98.9	93.5	21.6	74.4	51.5	99.0	**87.24**
	PN-Mix (2)	100.0	99.0	100.0	100.0	99.0	98.5	98.2	98.2	93.8	16.7	73.9	53.0	97.8	86.79
64K	FP16	100.0	100.0	100.0	99.5	98.0	98.5	98.4	99.8	87.5	0.6	68.8	50.0	96.8	84.44
	KIVI	99.0	99.5	94.5	98.0	91.0	22.5	91.9	96.4	86.8	1.8	65.6	48.5	90.6	75.85
	ChanMix	100.0	99.5	99.5	99.5	98.0	76.0	93.1	98.6	87.3	1.7	66.4	48.0	92.3	81.54
	Kitty	100.0	99.5	97.5	99.5	94.5	58.5	97.4	99.4	87.8	2.2	67.6	49.0	91.2	80.32
	PN-Mix (1)	100.0	100.0	100.0	99.5	97.0	98.0	97.4	99.5	88.2	0.9	68.9	50.5	94.1	**84.15**
	PN-Mix (2)	100.0	100.0	100.0	99.5	97.5	98.5	97.0	99.9	88.2	0.8	67.8	50.5	93.4	84.08

Table 13: **Accuracy on RULER across context lengths from 16K to 64K with Llama-3.1-8B-Instruct.** Tasks are the 13 official RULER categories, where NIAH denotes needle-in-a-haystack: single NIAH (S1–S3), multi-key NIAH (MK1–MK3), multi-value NIAH (MV), multi-query NIAH (MQ), frequent-word extraction (FWE), common-word extraction (CWE), question answering (QA1–QA2), and variable tracking (VT). All compressed configurations (i.e., every method except FP16) are matched to an average key-cache precision of 2.25–2.39 bits, while the value cache is kept in full precision. Bold indicates the best non-FP16 result in each column.

Configuration	Avg key bit	AIME24	
		Qwen3-8B	R1-Llama-8B
Full (FP16)	16.00	$76.33_{\pm 4.82}$	$48.33_{\pm 6.19}$
KIVI			
b=2, G=32	3.00	$72.33_{\pm 3.96}$	$42.67_{\pm 8.14}$
b=2, G=64	2.50	$68.00_{\pm 4.99}$	$35.00_{\pm 9.80}$
b=2, G=128	2.25	$64.00_{\pm 5.54}$	$28.67_{\pm 7.02}$
b=1, G=32	2.00	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
b=1, G=64	1.50	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
Kitty			
b_H=4, b_L=2, r=30%, G=128	2.92	$77.67_{\pm 4.96}$	$48.33_{\pm 7.19}$
b_H=4, b_L=2, r=10%, G=128	2.51	$73.33_{\pm 4.71}$	$46.67_{\pm 3.94}$
b_H=4, b_L=2, r=3.125%, G=128	2.39	$70.67_{\pm 5.93}$	$42.00_{\pm 5.62}$
b_H=3, b_L=1, r=30%, G=64	2.23	$40.00_{\pm 6.50}$	$1.33_{\pm 1.63}$
b_H=3, b_L=1, r=20%, G=64	2.03	$9.33_{\pm 2.49}$	$0.00_{\pm 0.00}$
b_H=3, b_L=1, r=10%, G=128	1.51	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
PN-Mix (Ours)			
b_H=3, b_L=2, r=70%, G=128	2.95	$76.67_{\pm 3.33}$	$48.33_{\pm 7.19}$
b_H=3, b_L=2, r=40%, G=64	2.90	$75.33_{\pm 4.52}$	$48.67_{\pm 5.62}$
b_H=4, b_L=1, r=30%, G=64	2.40	$75.00_{\pm 5.00}$	$48.33_{\pm 6.19}$
b_H=3, b_L=1, r=40%, G=64	2.29	$76.00_{\pm 4.67}$	$50.67_{\pm 5.54}$
b_H=3, b_L=1, r=40%, G=128	2.04	$74.67_{\pm 3.40}$	$47.33_{\pm 6.11}$
b_H=3, b_L=1, r=20%, G=64	1.89	$75.00_{\pm 3.07}$	$46.67_{\pm 3.65}$
b_H=3, b_L=1, r=10%, G=128	1.45	$67.66_{\pm 3.95}$	$40.33_{\pm 4.58}$

Table 14: **Configurations for the bit-rate Pareto frontier in Figure 3.** For each operating point, we report the selected hyperparameters and the corresponding AIME24 accuracy (%) with $b_V = 16$ fixed. Kitty and PN-Mix both preserve $N_s = 32$ sink tokens in FP16. *Full* denotes the FP16 KV-cache baseline.