# GRIK: GROUPED RELEVANCE-INTEGRATED KEY CACHE PRUNING FOR GROUPED-QUERY ATTENTION LLMS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

The key–value (KV) cache is a major memory bottleneck in long-context inference, and existing K-channel pruning does not shrink it under grouped-query attention (GQA): in the evaluated implementations, ThinK materializes K and V per query head ($3.20\times$ total KV) and SparK reconstructs a full-width cache ($1.00\times$). We identify two structural conditions, a mask shared across the query heads of each KV group and across all cached tokens, under which channel pruning yields a dense shared-key cache whose width equals the retained channel count. We introduce GRIK, a training- and calibration-free method that satisfies both conditions by construction and derives a prompt-adaptive mask per KV head from output-projection influence, key-projection norms, and attention energy. At 40% K-channel pruning, GRIK stores K at $0.60\times$ and total K+V at $0.80\times$ the bf16 baseline. Without token eviction, its LongBench average is within $0.23$ points of Full KV on three GQA backbones; with H2O eviction it is the best non-Full-KV method in 19 of 24 (model, budget, ratio) cells, and at approximately matched total-KV memory it improves over H2O alone by $1.55$–$1.62$ points. The reduction composes with other axes: adding GRIK to SnapKV, FP8, or int4 KV changes LongBench by at most $0.21$ points while cutting total KV to as low as $0.20\times$. On RULER, its eight exact-retrieval tasks average at least $99.0\%$ through 32K, although the 13-task average trails Full KV by $6.90$–$8.59$ points. In a fixed-memory H100 serving test at 50% pruning, the smaller cache admits $1.33\times$ more concurrent sequences and raises aggregate request throughput by $1.10\times$, with per-user decode latency $1.35\times$ that of the baseline at the higher concurrency. Code and configurations are available at `https://anonymous.4open.science/r/GRIK/`.

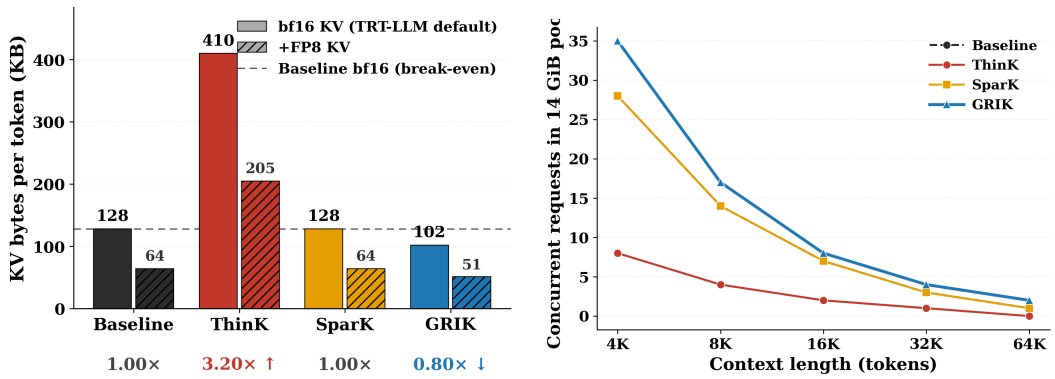
## 1 INTRODUCTION

Large language models (LLMs) have achieved strong performance across a wide range of real-world applications, but this progress has introduced a growing systems challenge: modern deployment increasingly requires long-context inference at scale (Bai et al., 2024; Hsieh et al., 2024). In autoregressive decoding, the dominant bottleneck is often no longer model parameters, but the key-value (KV) cache, which stores token-wise representations to avoid recomputation across decoding steps (Shazeer, 2019). Although essential for efficient generation, the KV cache grows linearly with both context length and batch size, and can quickly exceed the memory footprint of the model weights themselves (Kwon et al., 2023). For Llama-3-8B (Grattafiori et al., 2024), for example, the KV cache reaches 128 GiB at sequence length 32K and batch size 32, roughly $8\times$ the model's weight size and beyond the capacity of a single A100 GPU. Thus, the practical scalability of long-context LLMs is constrained by runtime KV memory rather than by model size alone.

Recent efficient LLMs widely adopt grouped-query attention (GQA) (Ainslie et al., 2023), which maps $H$ query heads to $H_{kv}$ shared key-value heads and reduces per-token KV storage from $H \cdot d$ to $H_{kv} \cdot d$. GQA leaves the per-head channel dimension $d$ untouched, so channel pruning is the natural next step. Under GQA, however, a channel mask reduces stored bytes only if it can be folded into the shared per-KV-head tensor. We formalize this as two conditions (§2.2): the mask must be shared by all query heads in a group (C1) and by all cached tokens (C2). Existing channel-pruning methods were designed without this constraint. ThinK (Xu et al., 2025) selects per-query-head masks



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




**(a)** Per-token KV memory.

**(b)** Concurrent capacity in a 14 GiB pool.

Figure 1: **KV storage and implied concurrent capacity on LLaMA-3-8B. (a)** Per-token total K+V memory at $r$=0.4. GRIK retains 77 of 128 K channels while leaving V unchanged, yielding 0.80× total KV storage. The plotted native ThinK implementation materializes K and V at query-head granularity, yielding approximately 3.20× total KV storage; SparK retains the full dense footprint; a group-averaged ThinK variant (ThinK-mean, §5.4) shares GRIK's 0.80× layout. Solid bars show bf16 storage and hatched bars show FP8 storage. **(b)** The number of sequences calculated to fit in a 14 GiB KV pool from these storage layouts. This panel shows capacity implied by cache size, rather than measured serving throughput.

(violating C1), and its native GQA implementation materializes K and V at query-head granularity, so the measured cache grows to 3.20× the GQA baseline; SparK (Liao et al., 2026) selects per-token masks (violating C2) and reconstructs a full-width tensor, so the measured cache stays at 1.00× (Figure 1). Both methods exhibit channel sparsity without realizing storage reduction.

We propose **GRIK** (Grouped Relevance-Integrated Key pruning), which treats channel pruning under GQA as mask selection inside the C1/C2-realizable space. GRIK assigns one prompt-adaptive mask per KV head, selected at prefill without training or calibration by combining three signals: the output-projection influence of each query head in the group, a static K-projection norm prior, and prompt-specific query/key energy. Pruning a fraction $r$ of channels then yields a dense K cache of shape $\mathbb{R}^{H_{kv} \times T \times k}$ by construction, and the reduction composes with token eviction and KV quantization because it acts on a third axis (feature width) rather than on token count or bytes per element.

**Contributions.** (1) We characterize the exact $k$-width shared-KV layout under GQA (C1/C2) and show that the two representative channel-pruning implementations fail it in complementary ways (§2.2–§3). (2) We introduce GRIK, a training- and calibration-free prompt-adaptive mask that satisfies C1/C2 by construction (§4). (3) With the storage layout held fixed, GRIK's score improves over uniform within-group aggregation on two backbones ($g$=4, 7), and each score component contributes (§5.4). (4) We evaluate the realized memory end to end: matched-total-memory quality, composition with SnapKV, FP8, and int4, and fixed-memory H100 serving (§5.3).

GRIK prunes K only, so its total K+V footprint at $r$=0.4 is 0.80×; we show that this reduction alone retains 1.25× more tokens at equal memory, raises fixed-memory serving throughput, and costs at most 0.23 LongBench points against the unpruned cache.

## 2 Preliminaries and Problem Setting

This section formalizes the KV cache structure of GQA-based LLMs (§2.1) and defines the structural conditions under which channel pruning translates into actual storage reduction (§2.2). Section 3 then examines how existing channel-pruning methods interact with these conditions on GQA models, and Section 4 presents GRIK as the design that satisfies them by construction.

Table 1: **Method comparison. (a)** Storage layout and KV-pool footprint. **(b)** Empirical quality vs. structural conditions C1, C2 (Definition 2.1).

**(a) Storage layout and KV pool**

| Method | Mask granularity | Mem. | Pool@64K |
|---|---|---|---|
| Full KV | — | 1.00× | 8.00 GiB |
| ThinK | per-query-head | 3.20× | 25.63 GiB |
| SparK | Per-token | 1.00× | 8.00 GiB |
| **GRIK** | **Per-KV-head** | **0.80×** | **6.41 GiB** |

**(b) Quality (LLaMA-3.1-8B, $r$=0.4) vs. C1, C2**

| Method | LB, H2O | LB, no evict | C1 | C2 |
|---|---|---|---|---|
| Full KV | 44.88 | 46.32 | ✓ | ✓ |
| ThinK | 42.79 | 46.34 | ✗ | ✓ |
| SparK | 39.94 | 33.50 | ✓ | ✗ |
| **GRIK** | **44.63** | **46.23** | ✓ | ✓ |

### 2.1 GQA AND KV CACHE STRUCTURE

Grouped-Query Attention (GQA) (Ainslie et al., 2023) maps $H$ query heads to $H_{\text{kv}}$ KV heads, with each KV head shared by $g = H/H_{\text{kv}}$ query heads. Modern LLM families including LLaMA-3 (Grattafiori et al., 2024), Mistral (Jiang et al., 2023), and Qwen-2.5 (Yang et al., 2024) adopt GQA as their default architecture, a design decision that reduces KV cache size by a factor of $g$.

For an input sequence of length $T$, the K cache at one layer is stored as $\mathbf{K} \in \mathbb{R}^{H_{\text{kv}} \times T \times d}$, where $d$ denotes the head dimension. Each KV head $h_{\text{kv}}$ maps token embeddings to key vectors through a projection matrix $W_k^{(h_{\text{kv}})} \in \mathbb{R}^{d \times d_{\text{model}}}$, and the query heads $h \in \mathcal{G}_{h_{\text{kv}}}$ in the group share the same $\mathbf{K}_{h_{\text{kv}}}$. Following scaled dot-product attention (Vaswani et al., 2017), the output of query head $h$ is

$$\mathbf{a}_h = \text{softmax}\left(\frac{\mathbf{q}_h \mathbf{K}_{h_{\text{kv}}}^\top}{\sqrt{d}}\right)\mathbf{V}_{h_{\text{kv}}}, \tag{1}$$

which is then transformed by the output projection $W_O^{(h)} \in \mathbb{R}^{d_{\text{model}} \times d}$ and added to the residual stream.

Throughout the paper we use the channel pruning ratio $r \in [0, 1)$ (the fraction of channels dropped) and the retained channel count $k = d - \text{round}(rd)$. The channel mask for KV head $h_{\text{kv}}$ is written $M_{h_{\text{kv}}} \in \{0, 1\}^d$ with $\|M_{h_{\text{kv}}}\|_0 = k$. A full notation table is provided in Appendix B.

### 2.2 STRUCTURAL CONDITIONS FOR AN EXACT $k$-WIDTH GQA CACHE

We characterize when channel pruning realizes a shared dense GQA key cache whose width equals the retained channel count $k$. This specific layout gives a predictable reduction in stored K bytes.

**Definition 2.1** (Exact $k$-Width Shared-KV Layout)**.** *Let $\{M_h^{(t)} \subseteq [d] : h \in [H_q], t \in [T]\}$ be channel supports of size $k$ each. We seek one packed dense K tensor of shape $\mathbb{R}^{H_{\text{kv}} \times T \times k}$ with one fixed channel-index map per KV head and no additional per-query-head or per-token K storage. This exact layout is possible if and only if both conditions hold:*

- ***(C1) Group consistency:*** *all query heads sharing a KV head use the same support at each token, i.e., $M_h^{(t)} = M_{h'}^{(t)}$ for all $h, h' \in \mathcal{G}_{h_{\text{kv}}}$ and $t \in [T]$.*
- ***(C2) Token consistency:*** *each query head uses a support that is constant across tokens, i.e., $M_h^{(t)} = M_h^{(t')}$ for all $h \in [H_q]$ and $t, t' \in [T]$.*

**Remarks.** If two size-$k$ masks in a group (C1) or across tokens (C2) differ, their union exceeds $k$, so no fixed $k$-width index map can represent them (Appendix C.2). A wider union-packed or indexed layout could still save some memory; C1/C2 are therefore sufficient for the exact $k$-width layout, and the 3.20× and 1.00× figures in Figure 1 describe what the released ThinK and SparK implementations materialize. Table 1 pairs the two failure modes against GRIK on a single grid: panel (a) reports the realized K-cache pool footprint at long context, and panel (b) reports the corresponding accuracy alongside which structural conditions each method satisfies.

## 3 DIAGNOSIS: EXISTING CHANNEL PRUNING FAILS UNDER GQA

We examine how the two representative channel-pruning methods behave under GQA, using C1 and C2 to separate nominal sparsity from realized storage.

**ThinK.** ThinK (Xu et al., 2025) selects per-query-head, query-adaptive channel masks from attention magnitude. Under MHA each query head owns its K projection, so per-head masks are storage-compatible. Under GQA the $g$ query heads sharing a KV head obtain different masks, violating C1; the released implementation resolves this by materializing K and V at query-head granularity, which yields $2.41\times$ K-only and $3.20\times$ total-KV storage at $r{=}0.4$ on LLaMA-3-8B (Figure 1, §5.3).

**SparK.** SparK (Liao et al., 2026) selects per-token masks and recovers pruned entries by sampling from per-channel statistics. Token-varying supports violate C2, and the released implementation reconstructs a full-shaped dense K tensor, so its measured footprint equals the dense baseline (Appendix C.2). The recovery noise also compounds with group size: on Qwen-2.5 ($g{=}7$) SparK collapses to a LongBench average near 11 (§5.2).

**Table 1.** Panel (a) reports the total K+V pool at context length $65{,}536$, batch 1, bf16, on LLaMA-3-8B-Instruct at $r{=}0.4$ ($d_{\text{keep}}{=}77$); the layout formula matches the TRT-LLM 1.0.0 allocator within $0.4\%$ (Appendix L). Panel (b) reports LongBench 16-task averages on LLaMA-3.1-8B-Instruct at $r{=}0.4$ under H2O eviction ($B{=}1024$) and without eviction, so that the channel effect is visible both with and without a token policy. GRIK is the only method that satisfies both conditions, stores total KV at $0.80\times$, and stays within $0.3$ LongBench points of Full KV.

**Positioning.** ThinK and SparK succeed on the axes they were designed for (MHA channel compression and accuracy-preserving unstructured sparsity). GRIK targets a different axis: dense-tensor storage reduction under GQA. C1 and C2 reduce that problem to choosing one mask per KV head, shared across the group and across tokens; the next section gives a training-free rule for choosing it.

## 4 GRIK: GROUPED RELEVANCE-INTEGRATED KEY PRUNING

The structural analysis in §2 reduces channel pruning under GQA to a single object: a per-KV-head mask $M_{h_{\text{kv}}} \in \{0,1\}^d$ shared by all query heads in the group and applied uniformly across tokens. GRIK is the simplest training- and calibration-free instantiation of this constraint. The channel score combines three measurable signals (the output-projection influence $\alpha_h$ of each query head, the static prior $w_k$ from the K-projection weight norms, and the prompt-specific dynamic signal $E_q \cdot E_k$ from attention energy) into a single per-KV-head score:

$$s^{(t)}_{h_{\text{kv}}}[d'] \;=\; \underbrace{w_k[h_{\text{kv}}, d']}_{\text{static prior}} \cdot \sum_{h \in \mathcal{G}_{h_{\text{kv}}}} \alpha^{(t)}_h \cdot \underbrace{E_q[h, d'] \cdot E_k[h_{\text{kv}}, d']}_{\text{dynamic signal}}. \tag{2}$$

The mask is $M_{h_{\text{kv}}} = \text{Top-}k(s^{(N)}_{h_{\text{kv}}}, k)$ with $k = d - \text{round}(rd)$; for $d{=}128$, $r{=}0.4$ gives $k{=}77$.

**Signals.** The static prior is the K-projection row norm normalized by its within-head mean, $w_k[j,c] = \|W^{(j)}_k[c,:]\|_2 \,/\, \frac{1}{d}\sum_{c'} \|W^{(j)}_k[c',:]\|_2$, computed once per checkpoint. The dynamic signal is the per-channel energy of the prompt on each side of the dot product, $E_q[h,c] = \mathbb{E}_t[q^2_{h,t,c}]$ and $E_k[j,c] = \mathbb{E}_t[k^2_{j,t,c}]$, so $E_q \cdot E_k$ is the expected contribution of channel $c$ to the attention logits of head $h$ on this prompt. The head weight $\alpha^{(t)}_h$ is the normalized output-projection influence of head $h$ under the current mask: with $\mathbf{a}^{(t)}_h(u)$ the attention output at observation query $u$, $\phi^{(t)}_h = \frac{1}{|\mathcal{O}_t|}\sum_u \|W^{(h)}_O \mathbf{a}^{(t)}_h(u)\|_2$ and $\alpha^{(t)}_h = \phi^{(t)}_h / \sum_{h' \in \mathcal{G}_j} \phi^{(t)}_{h'}$ (Appendix D). Heads that write more into the residual stream therefore carry more weight in the group decision, following the observation that head contributions are highly unequal (Michel et al., 2019; Voita et al., 2019).

The score is multiplicative rather than additive for the same reason as weight-times-activation pruning (Sun et al., 2024; Lin et al., 2024): a channel should survive only if it is both intrinsically

strong in $W_k$ and active on the current prompt. An additive form can rank a channel highly on one factor alone; the product acts as a joint gate, and §5.4 shows that removing either factor costs accuracy.

**Iterative refinement.** GRIK uses an $N$-step schedule: at step $t$ it recomputes attention and $\alpha_h^{(t)}$ under the current mask, combines these weights with the fixed prompt energies, and selects the top $k_t$ channels from all $d$; previously dropped channels may re-enter (step-to-step Jaccard falls from $0.990$ to $0.933$ over the schedule). A 32-query observation window is split across the $N$ steps. Refinement attention scales the active dot product by $\sqrt{k_{t-1}}$ (Appendix C.4); decoding uses the original $\sqrt{d}$ scale. We report two modes: $N{=}4$ (latency-oriented, recommended) and $N{=}32$ (quality-oriented, used in the main tables); the two differ by at most $0.32$ LongBench points (§5.4).

**Does one shared mask hurt individual heads?** A shared mask is only reasonable if the query heads in a group want similar channels. We measured this on LLaMA-3-8B over 12 LongBench prompts, all 32 layers and all 8 KV groups: selecting each query head's own top-77 channels independently gives a mean within-group Jaccard of $0.70$ versus $0.43$ at chance, and the final GRIK mask covers $87\%$ of each head's own top-77 set on average and $81\%$ for the worst head. Aggregation therefore preserves most of what every head would have chosen alone rather than sacrificing one head for the group average.

**Complexity.** Mask selection happens once at prefill. Prompt energies cost $O(HTd)$ and top-$k$ selection $O(NH_{\mathrm{kv}}d\log d)$; each refinement step additionally recomputes attention on its observation queries, so the measured overhead depends on $T$ and $N$ (Appendix G). Decoding reads an asymmetric dense K/V cache with K width $k$ and V width $d$ (§5.3).

# 5 Experiments

We answer four questions: does K-channel pruning cost quality when token eviction is removed and when it is present (§5.2); does the realized $0.80\times$ footprint buy quality at equal memory, compose with other compression axes, and translate into serving capacity (§5.3); which parts of the score matter (§5.4); and how retrieval-heavy and reasoning workloads behave (§5.5). Appendix Figure 2 plots accuracy and K-cache memory against the KV budget on Qwen-2.5-7B and LLaMA-3.1-8B.

## 5.1 Settings

We evaluate GRIK on three GQA backbones (LLaMA-3-8B, LLaMA-3.1-8B, Qwen-2.5-7B; $g \in \{4, 4, 7\}$) (Grattafiori et al., 2024; Yang et al., 2024) on LongBench-v1 (Bai et al., 2024) (16 tasks) and RULER (Hsieh et al., 2024) ($\{4, 8, 16, 32\}$K). Channel-pruning baselines are ThinK (Xu et al., 2025) and SparK (Liao et al., 2026). "Full KV" denotes the channel-unpruned cache; in the budgeted LongBench tables all methods, including Full KV, share the same H2O (Zhang et al., 2023) eviction prefix at token budget $B \in \{128, 512, 1024, 2048\}$ with recent window $w{=}32$, so Full KV there means *unpruned K/V under the same eviction*. "No eviction" denotes a budget larger than every evaluated context, where the only intervention is the K width. Pruning ratios are $r \in \{0.4, 0.6\}$; RULER uses a 200K budget without eviction at $r{=}0.4$ (primary) and $r{=}0.7$ (stress test). Details are in Appendix E.

## 5.2 Long-Context Benchmark

Table 2 reports the $N{=}32$ LongBench-v1 16-task averages across three backbones, two ratios, and four budgets (24 cells; per-task tables in Appendix H). GRIK is the strongest non-Full-KV method on $19/24$ cells and within $0.6$ points of the leader on $4$ of the remaining $5$; the five losses are all to ThinK at $B{=}128$ or on Qwen-2.5 at $r{=}0.6$. On LLaMA-3.1-8B at $r{=}0.4$, GRIK trails Full KV by at most $0.27$ points at every budget while ThinK loses $0.65$–$2.09$ and SparK $1.65$–$7.26$. SparK collapses on Qwen-2.5 (AVG 11 at every $B$): per-token recovery in a $g{=}7$ group amplifies channel-removal noise (Appendix C.2), and at $r{=}0.6$ it degrades further on every backbone. Paired bootstrap intervals over the 16 tasks (Appendix I) give $P(\text{GRIK} > \text{SparK}){=}1.00$ in every cell and show GRIK statistically indistinguishable from Full KV on LLaMA-3.1-8B at $r{=}0.4$ (gap CI within $\pm 0.5$ at every

Table 2: LongBench-v1 16-task average across three GQA backbones, four KV budgets, pruning ratios $r$=$\{0.4, 0.6\}$ (H2O eviction). **Bold** marks the best non-Full-KV cell per (model, budget, ratio). Rightmost column: asymptotic K-cache ratio vs. Full KV (GRIK = $1-r$, SparK = $1\times$, ThinK = $g\cdot(1-r)$). Per-task tables in Appendix H; no-eviction control in Table 9; paired bootstrap confidence intervals over 16 tasks ($N$=10,000) in Appendix I.

| Model | Method | $B$=128 | $B$=512 | $B$=1024 | $B$=2048 | K-cache |
|---|---|---|---|---|---|---|
| LLaMA-3-8B-Inst. | Full KV (reference)* | 35.38 | 37.23 | 38.70 | 39.59 | 1.00× |
| | ThinK† ($r$=0.4) | 35.63 | 37.39 | 39.00 | 40.05 | 2.40× |
| | SparK ($r$=0.4) | 32.73 | 33.92 | 34.68 | 34.83 | 1.00× |
| | **GRIK** ($r$=0.4) | **35.92** | **38.63** | **40.12** | **41.30** | **0.60×** |
| | ThinK† ($r$=0.6) | **35.06** | 36.44 | 37.58 | 38.51 | 1.60× |
| | SparK ($r$=0.6) | 27.97 | 26.62 | 24.11 | 21.92 | 1.00× |
| | **GRIK** ($r$=0.6) | 35.00 | **37.45** | **38.45** | **39.24** | **0.40×** |
| LLaMA-3.1-8B-Inst. | Full KV (reference) | 39.87 | 43.38 | 44.88 | 45.73 | 1.00× |
| | ThinK ($r$=0.4) | 39.22 | 41.56 | 42.79 | 44.30 | 2.40× |
| | SparK ($r$=0.4) | 38.22 | 39.94 | 39.94 | 38.47 | 1.00× |
| | **GRIK** ($r$=0.4) | **39.60** | **43.17** | **44.63** | **45.46** | **0.60×** |
| | ThinK ($r$=0.6) | 39.00 | 40.89 | 42.38 | 43.65 | 1.60× |
| | SparK‡ ($r$=0.6) | 34.06 | 29.86 | 25.12 | 20.65 | 1.00× |
| | **GRIK** ($r$=0.6) | **39.31** | **42.07** | **43.54** | **44.62** | **0.40×** |
| Qwen2.5-7B-Inst. | Full KV (reference) | 39.58 | 42.35 | 43.79 | 44.89 | 1.00× |
| | ThinK ($r$=0.4) | **39.41** | 42.16 | 43.68 | 45.13 | 4.20× |
| | SparK ($r$=0.4) | 10.73 | 11.41 | 11.49 | 11.28 | 1.00× |
| | **GRIK** ($r$=0.4) | 38.88 | **42.53** | **43.98** | **45.29** | **0.60×** |
| | ThinK ($r$=0.6) | **39.40** | 41.93 | **43.64** | **44.92** | 2.80× |
| | SparK ($r$=0.6) | 10.55 | 10.93 | 10.98 | 10.86 | 1.00× |
| | **GRIK** ($r$=0.6) | 38.27 | **42.05** | 43.49 | 44.73 | **0.40×** |

Table 3: LongBench-v1 16-task average at approximately matched total K+V storage on LLaMA-3-8B-Instruct. GRIK uses $r$=0.4, $N$=4, and H2O eviction; H2O-only uses full-width K and V. Difference: GRIK minus H2O-only.

| H2O $B$ | H2O AVG | GRIK $B$ | GRIK AVG | $\Delta$ |
|---|---|---|---|---|
| 512 | 37.47 | 640 | 39.09 | +1.62 |
| 1024 | 38.64 | 1280 | 40.19 | +1.55 |

budget). Per task, the largest separation from ThinK and SparK appears on retrieval-style subtasks (NarrativeQA, Qasper, HotpotQA, PassageRetrieval), while summarization and code completion are nearly unaffected by any method; SparK's losses at $r$=0.6 concentrate on the same retrieval-heavy tasks (Appendix H).

With eviction disabled, GRIK differs from Full KV by at most 0.23 LongBench points across the three backbones (Table 9), and by at most 0.79 points at $r$=0.6. This control separates the effect of reducing the K width from the effect of retaining a different token set.

At the asymptotic K+V ratio $(1 + 77/128)/2 \approx 0.8008$, GRIK can retain approximately $1.25\times$ as many tokens as H2O-only. At the two approximately equal-storage operating points in Table 3, this additional context improves the 16-task average by 1.55–1.62 points. Because the most recent $w$=32 tokens retain full-width K, the modeled GRIK storage at $B$=640/1280 is approximately $1.013\times$/$1.007\times$ that of H2O-only at $B$=512/1024, respectively, before implementation overhead; the budgets are therefore approximate rather than exact memory matches. The H2O-only scores come from separate matched-budget runs and should not be identified with the cited Full-KV row

Table 4: Composition of GRIK ($r$=0.4, $N$=32) with token eviction and KV quantization on LLaMA-3-8B, LongBench AVG. Total KV: asymptotic data-byte ratio vs. the bf16 unpruned cache at the same budget.

| Axis | Configuration | $B$ | AVG | Total KV |
|---|---|---|---|---|
| Eviction | SnapKV alone | 1024 | 40.97 | 1.00× |
| | GRIK + SnapKV | 1024 | 41.16 | 0.80× |
| FP8 | GRIK + H2O, bf16 | 2048 | 41.30 | 0.80× |
| | GRIK + H2O, FP8 | 2048 | 41.09 | 0.40× |
| int4 | GRIK + H2O, bf16 | 1024 | 40.12 | 0.80× |
| | GRIK + H2O, int4 | 1024 | 40.17 | 0.20× |

Table 5: Fixed-memory H100 serving. One H100, 2,048-token input, 128-token output, same scheduler, GPU-memory utilization 0.85, fused FA3 path; GRIK at $r$=0.5 (K/V widths 64/128). GRIK admits 1.33× more concurrent sequences, so per-user rows are at different concurrency, not a matched batch.

| Metric | Full KV | GRIK |
|---|---|---|
| Allocatable KV blocks | 25,666 | 34,221 |
| Aggregate requests/s | 8.20 | 8.99 |
| Output tokens/s/GPU | 1,050 | 1,151 |
| Per-user tokens/s | 95.2 | 70.4 |
| Decode latency (ms/token) | 10.50 | 14.21 |
| Time to first token (ms) | 80.6 | 84.0 |

in Table 2. This $N$=4 capacity comparison is distinct from the $N$=32 equal-token-budget quality comparisons in Table 2.

### 5.3 Realized Memory: Matched Budgets, Composition, and Serving

At a fixed retained-token count, GRIK changes only the K width from $d$ to $k$, so its asymptotic ratios are $k/d$ for K and $(k/d+1)/2$ for total K+V: 0.60× and 0.80× at $r$=0.4 on LLaMA-3-8B ($k$=77). SparK retains 1.00×. The native ThinK-on-GQA layout stores K and V per query head, giving $gk/d \approx 2.41\times$ for K and $g(k/d+1)/2 \approx 3.20\times$ total. These formulas match the materialized cache within mask overhead (Appendix F, Table 12) and the TRT-LLM allocator within 0.4% (Appendix L). At equal context length a 14 GiB pool therefore holds 1.25× as many GRIK sequences as Full KV (Figure 1b).

**Quality at equal memory.** Because GRIK changes tensor shape rather than token count or bytes per element, the deployment question is whether the 0.80× footprint buys quality when total memory is fixed. Table 3 shows that it does: at approximately equal total K+V bytes, GRIK with H2O retains 1.25× more tokens and scores 1.55–1.62 points above H2O alone.

**Composition.** Table 4 tests same-budget controls on LLaMA-3-8B. Replacing H2O by SnapKV and adding GRIK improves over SnapKV alone by 0.19 points at 0.80× total KV. Adding FP8 or KIVI-style asymmetric int4 KV quantization on top of GRIK changes the average by −0.21 and +0.05 points while reducing total KV to 0.40× and 0.20× of the bf16 unpruned baseline, so channel width, token count, and precision reduce independent factors of the footprint and their savings multiply.

**Fixed-memory serving.** Table 5 measures a fused FA3 asymmetric-K/V path at $r$=0.5, the aligned point supported by the current kernel. The smaller cache raises allocatable KV blocks by 1.33× and aggregate requests/s and tokens/s/GPU by 1.10× under the same GPU-memory budget; time to first token changes by 4%. At the higher concurrency, per-user tokens/s is 0.74× and decode latency 1.35× the baseline, so the gain is aggregate capacity when KV memory bounds concurrency; a kernel specialized for the asymmetric width is the direct route to per-user latency. Quality at this ratio matches $r$=0.4: LongBench 35.44/38.35/39.67 at $B$=128/512/1024, RULER-13 86.09/86.05 at 4K/8K, NIAH 99.8/99.6. Prefill overhead at $N$=32 is +148% at 4K and +53% at 32K on LLaMA-3-8B; one-shot scoring costs +8–14% and $N$=4 lies between (Appendix G).

### 5.4 Ablation

GRIK's score adds three ingredients on top of the shared-mask layout: (i) $\alpha_h$-weighted within-group aggregation, (ii) the static $\|W_k\|$ prior (Sun et al., 2024), and (iii) iterative refinement. Holding the layout fixed, the natural reference with none of them is **ThinK-mean**: ThinK's per-query-head score averaged uniformly within each KV group, which materializes exactly GRIK's $H_{kv} \times T \times k$ cache.

Table 6: **Score-form ablation.** GRIK vs. ThinK-mean on LLaMA-3-8B, LongBench 16-task AVG. $\Delta$: GRIK − ThinK-mean.

| **Method** | $B$=128 | $B$=512 | $B$=1024 | $B$=2048 |
|---|---|---|---|---|
| $r = 0.4$ | | | | |
| ThinK-mean | 35.45 | 36.96 | 38.11 | 39.48 |
| **GRIK (Ours)** | **35.92** | **38.63** | **40.12** | **41.30** |
| $\Delta$ | +0.47 | +1.67 | +2.01 | +1.82 |
| $r = 0.6$ | | | | |
| ThinK-mean | 34.91 | 36.30 | 37.40 | 38.30 |
| **GRIK (Ours)** | **35.00** | **37.45** | **38.45** | **39.24** |
| $\Delta$ | +0.09 | +1.15 | +1.05 | +0.94 |

Table 7: **Component ablation.** LongBench 16-task AVG at $r$=0.4 with the per-KV-head layout fixed. Top: LLaMA-3-8B. Bottom: ThinK-mean vs. GRIK on Qwen-2.5-7B ($g$=7).

| **Variant (LLaMA-3-8B)** | $B$=128 | $B$=1024 |
|---|---|---|
| GRIK, full score ($N$=32) | **35.92** | **40.12** |
| w/o static prior $w_k$ | 35.46 | 39.81 |
| w/o $\alpha_h$ (uniform mean) | 35.45 | 38.11 |
| $N$=4 | 35.60 | 39.92 |
| $N$=1 (one-shot) | — | — |
| **Qwen-2.5-7B** | $B$=512 | $B$=1024 |
| ThinK-mean | 42.12 | 43.49 |
| **GRIK** | **42.53** | **43.98** |

Table 8: Primary RULER-13 result on LLaMA-3.1-8B-Instruct at $r$=0.4 (60% of K channels kept), KV budget 200K. Token eviction does not activate. Difference is GRIK minus Full KV in points; the final row averages the eight needle-in-a-haystack (NIAH) tasks within GRIK.

| Method / metric | 4K | 8K | 16K | 32K |
|---|---|---|---|---|
| Full KV | 95.10 | 93.10 | 90.20 | 86.00 |
| GRIK, $r$=0.4 | 86.51 | 85.96 | 83.30 | 77.70 |
| Difference | −8.59 | −7.14 | −6.90 | −8.30 |
| GRIK, 8 NIAH tasks | 99.9 | 99.6 | 99.6 | 99.0 |

Table 6 compares the two on LLaMA-3-8B across all $(B, r)$ cells; Table 7 removes each ingredient in turn and repeats the ThinK-mean comparison on Qwen-2.5 ($g$=7).

**Score form.** GRIK outperforms ThinK-mean in every cell on LLaMA-3-8B, by +0.09 to +2.01 points, with the gap widening as $B$ grows (paired-bootstrap $P$=1.00 at $B \geq 512$, Appendix I), and by +0.41/+0.49 on Qwen-2.5 at $B$=512/1024, so the gain is not specific to $g$=4. Removing the static prior costs 0.31–0.46 points; replacing $\alpha_h$ by a uniform group mean costs 0.47 points at $B$=128 and 2.01 at $B$=1024, so head weighting matters most when more context is retained. The multiplicative form follows the weight-times-activation rule of post-training compression (Sun et al., 2024; Lin et al., 2024): a channel is kept when it is both intrinsically strong in $W_k$ and active on the prompt. Alternative dynamic terms underperform ($K^2$-only −0.24; qsnr −0.02 to −0.17; Appendix J).

**Refinement depth.** The refinement-only contribution saturates quickly (+0.05 at $N$=1, +0.08 at $N$=4, +0.10 at $N$=32 in the controlled sweep). The complete $N$=4 configuration is within 0.14–0.32 points of $N$=32 at every budget (Table 7, $B$=128 and 1024; 38.49 vs. 38.63 at $B$=512, 41.14 vs. 41.30 at $B$=2048), while one-shot scoring adds 8–14% prefill time and $N$=32 adds 53–148% (Appendix G). We therefore recommend $N$=4 and report $N$=32 as the quality upper bound.

**Layout dominates score.** All score variants here lie within 2 points of each other, whereas the evaluated ThinK layout uses 3.20× total KV and GRIK 0.80× at the same retained-token count, a 4× gap (§5.3). The layout decides whether memory is saved; the score decides how much quality is kept.

### 5.5 Retrieval Stress Test and General Reasoning

Table 8 reports the primary $r$=0.4 setting without eviction: the 13-task average trails Full KV by 6.90–8.59 points across 4K–32K, and the gap does not grow with length (8.59 at 4K, 8.30 at 32K). The eight needle-in-a-haystack tasks average 99.0–99.9%, so exact retrieval is preserved; the loss is concentrated in global aggregation, most sharply common-word extraction (99.0 at 4K to 4.2 at

32K; per task in Appendix K). Frequent-word extraction and the two QA tasks change gradually with length, and variable tracking is the other sensitive task; the pattern is that operations requiring a global count over the whole context suffer from a narrower key, whereas operations that locate a specific span do not. At the aggressive $r$=0.7 (Table 21), GRIK falls to 54.24–61.52, ThinK to 36.40–58.30, and SparK below 7 with all eight NIAH tasks at zero, so the stable shared layout also matters for retrieval under heavy pruning.

**General reasoning.** On GSM8K (8-shot chain-of-thought, LLaMA-3.1-8B, the 234 questions completed by all methods), Full KV scores 77.8, SparK 75.2 at 1.00× K storage, GRIK 73.5 at 0.60×, ThinK 73.1 at 3.20×, and ThinK-mean 71.8 at 0.60×: GRIK is the best method that realizes the 0.60× dense cache and matches per-query-head ThinK at a fraction of the storage. On GSM-Infinite (1–30 operations, no eviction), where the model must carry a growing chain of variable assignments through the cache, GRIK matches Full KV (0.337 vs. 0.329 EM; 0.641 vs. 0.611 F1). The retained 60% of K channels therefore preserves the information needed for multi-step generation as well as for retrieval.

## 6 Related Work and Positioning

KV-cache compression acts along three largely orthogonal axes. Token eviction reduces the number of retained or accessed tokens through heavy-hitter, recent-window, retrieval-head, or query-aware policies (Zhang et al., 2023; Xiao et al., 2024; Li et al., 2024; Xiao et al., 2025; Tang et al., 2024); KV quantization reduces bytes per element without changing tensor shape (Liu et al., 2024; Hooper et al., 2024); channel pruning reduces the per-token feature width. GRIK sits on the third axis and composes with the other two (§5.3). Within channel pruning, ThinK (Xu et al., 2025) uses per-query-head masks designed for MHA and SparK (Liao et al., 2026) uses per-token masks with recovery; §3 shows that neither realizes a smaller dense GQA cache. LeanK (Zhang et al., 2025) is the closest design point: it also uses one mask per KV head, but learns a static mask offline with training and a dedicated kernel. GRIK is the training- and calibration-free, prompt-adaptive counterpart, and the two address different deployment constraints: a fixed mask amortized over a training run versus a per-prompt mask available for any checkpoint without data. The two are also compatible in principle, since a learned prior could replace $w_k$ in Equation 2 while the prompt-dependent term is kept. Ada-KV (Feng et al., 2025) and Twilight (Lin et al., 2025) make related observations about head-level heterogeneity under GQA on the token and attention-sparsity axes, respectively, but neither changes the stored feature width. Head-redundancy analyses (Michel et al., 2019; Voita et al., 2019) and the weight-times-activation rule (Sun et al., 2024; Lin et al., 2024) motivate the score (fuller survey in Appendix A).

## 7 Conclusion

We have shown that channel pruning reduces KV memory under grouped-query attention only when the channel mask is shared across the query heads of a group and across cached tokens, and that the two representative channel-pruning methods do not satisfy this requirement in their released form, leaving the measured cache at 3.20× or 1.00× of the baseline. Building on this observation, we proposed GRIK, a training- and calibration-free method that derives one prompt-adaptive mask per KV head and therefore yields a dense $H_{kv} \times T \times k$ key cache by construction. At $r$=0.4, GRIK stores K at 0.60× and total KV at 0.80×, remains within 0.23 LongBench points of the unpruned cache on three GQA backbones, preserves exact retrieval on RULER at 99% or above through 32K, and composes with SnapKV, FP8, and int4 quantization. The realized footprint translates into 1.55–1.62 LongBench points at equal memory and a 1.10× increase in aggregate H100 serving throughput under a fixed memory budget. Since GRIK introduces no parameters and leaves the decoding path unchanged apart from a narrower key tensor, it can be deployed on existing paged-attention serving stacks without modification. As the structural conditions constrain the mask rather than the model, the same construction extends naturally to other shared-KV settings, including per-expert masks in mixture-of-experts layers, per-window masks in sliding-window attention, and latent-level masks in multi-head latent attention, which we leave to future work.

### AI USE STATEMENT

During this revision, we used OpenAI ChatGPT (Codex) to check algebra and cache-memory calculations, examine assumptions in mathematical claims, compare reported numbers with existing experiment records, and help interpret the scope of the results. It also provided feedback on the presentation of the methodology and assisted with editing, organization, and related-literature checks. No new experimental results were generated by the tool during this revision. The authors take responsibility for all final claims and AI-assisted content.

## REFERENCES

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

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

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. LLM.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S. Kevin Zhou. Ada-KV: Optimizing KV cache eviction by adaptive budget allocation for efficient LLM inference. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

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

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

Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

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

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

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th ACM Symposium on Operating Systems Principles (SOSP)*, 2023.

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

Huanxuan Liao, Yixing Xu, Shizhu He, Guanchen Li, Xuanwu Yin, Dong Li, Emad Barsoum, Jun Zhao, and Kang Liu. SparK: Query-aware unstructured sparsity with recoverable KV cache channel pruning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2026.

Chaofan Lin, Jiaming Tang, Shuo Yang, Hanshuo Wang, Tian Tang, Boyu Tian, Ion Stoica, Song Han, and Mingyu Gao. Twilight: Adaptive attention sparsity with hierarchical top-$p$ pruning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. Spotlight.

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

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

Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

NVIDIA. TensorRT-LLM: A TensorRT toolbox for optimized large language model inference, 2024. Software, version 1.0.0, `https://github.com/NVIDIA/TensorRT-LLM`.

Victor Sanh, Thomas Wolf, and Alexander M. Rush. Movement pruning: Adaptive sparsity by fine-tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Noam Shazeer. Fast transformer decoding: One write-head is all you need, 2019. Multi-Query Attention; widely cited as the precursor to GQA.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. QUEST: Query-aware sparsity for efficient long-context LLM inference. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019.

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

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

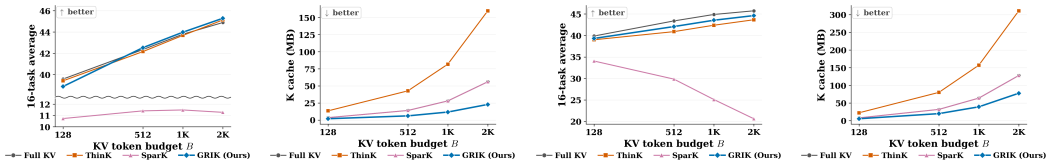
Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. DuoAttention: Efficient long-context LLM inference with retrieval and streaming heads. In *International Conference on Learning Representations (ICLR)*, 2025.

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

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Yike Zhang, Zhiyuan He, Huiqiang Jiang, Chengruidong Zhang, Yuqing Yang, Jianyong Wang, and Lili Qiu. Leank: Learnable k cache channel pruning for efficient decoding. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 31110–31125, 2025.

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




(a) LongBench-AVG (b) K-cache memory (c) LongBench-AVG (d) K-cache memory

Figure 2: **Accuracy and K-cache memory vs. KV budget $B$.** GRIK is the only method whose accuracy tracks Full KV while its K-cache stays below the Full KV reference at every $B$. The native ThinK implementation stores query-head K/V tensors, while SparK reconstructs a full-width K tensor (Appendix C.2); ratios depend on the chosen backbone and budget.

Table 9: Channel-only LongBench-v1 16-task average at $r$=0.4 with no token eviction. The KV budget exceeds each evaluated context, so the Full-KV comparison isolates K-channel pruning. Differences are GRIK minus Full KV in points.

| Model | Full KV | GRIK | Difference |
|---|---|---|---|
| LLaMA-3-8B-Instruct | 41.86 | 42.09 | +0.23 |
| LLaMA-3.1-8B-Instruct | 46.32 | 46.23 | −0.09 |
| Qwen2.5-7B-Instruct | 47.34 | 47.20 | −0.14 |

## A Related Work

KV-cache compression organizes along three orthogonal axes (token eviction, channel pruning, and quantization) that reduce different parts of the cache footprint and compose multiplicatively. We locate GRIK on the channel-pruning axis and discuss its relationship to each, then connect to the broader pruning literature.

**Token eviction.** H2O (Zhang et al., 2023) accumulates attention scores in a sliding window and discards low-scoring past tokens; SnapKV (Li et al., 2024) refines the eviction policy with a structured prior over recent tokens; Ada-KV (Feng et al., 2025) allocates per-head budgets adaptively to honor the heterogeneity of attention concentration across heads. StreamingLLM (Xiao et al., 2024) keeps the initial "attention sink" tokens together with a sliding window of recent tokens to enable infinite-length streaming, and DuoAttention (Xiao et al., 2025) extends head-level heterogeneity by partitioning heads into "retrieval" (full cache) and "streaming" (compressed) groups. Quest (Tang et al., 2024) dynamically selects top-$K$ critical KV pages per query rather than evicting them. These methods reduce the number of stored or accessed tokens rather than per-token feature dimension and are therefore complementary to channel pruning. We adopt an H2O-style eviction prefix throughout and apply channel pruning on top of the retained tokens; the two effects compose multiplicatively in the storage formula (§5.3). At the systems layer, vLLM (Kwon et al., 2023) manages the KV cache through paged-attention to eliminate fragmentation; integrating GRIK with such serving stacks is a natural deployment direction.

**Channel pruning.** ThinK (Xu et al., 2025) introduced channel pruning for MHA architectures with per-query-head masks based on attention magnitude. SparK (Liao et al., 2026) uses unstructured per-token masks and recovers dropped entries by sampling from a per-channel distribution. GRIK occupies the same axis but differs along two dimensions. First, GRIK formulates channel pruning as a structurally constrained selection problem under GQA (Ainslie et al., 2023) (which generalizes the multi-query attention (Shazeer, 2019) reduction to intermediate group sizes; Definition 2.1), yielding storage reduction *by construction* rather than as a side effect of the scoring rule; this distinguishes it from ThinK and SparK, which target different design axes and forfeit GQA's storage gain in distinct ways (§3). Second, GRIK is training- and calibration-free and decides masks per prompt; this distinguishes it from approaches that fix a static channel mask offline through calibration or fine-tuning. Related observations on the interaction between GQA and head-level compression have appeared in recent work (Feng et al., 2025; Lin et al., 2025), and broader analyses of transformer

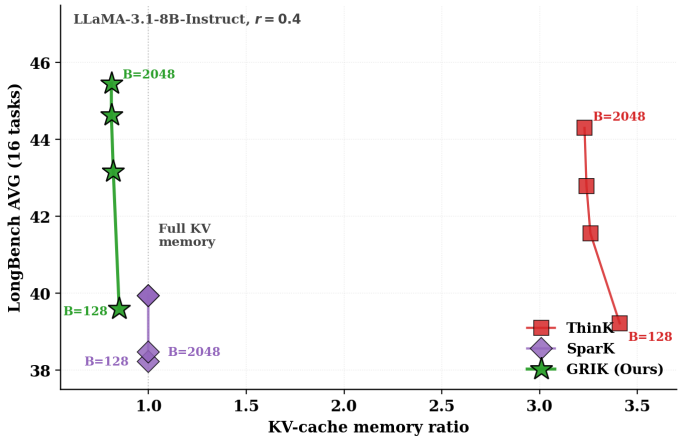



Figure 3: **Memory–accuracy Pareto frontier.** LLaMA-3.1-8B-Instruct, $r$=0.4. GRIK lies upper-left of ThinK and SparK at every budget $B \in \{128, 512, 1024, 2048\}$.

redundancy at the head level (Michel et al., 2019; Voita et al., 2019) motivate the heterogeneous treatment of channels within heads we adopt.

**Quantization.** A complementary axis to KV-cache compression is reducing per-element bit precision. Weight-only quantization (GPTQ (Frantar et al., 2023), AWQ (Lin et al., 2024)) reduces parameter storage to 3–4 bits while preserving generation quality. Activation-aware methods such as SmoothQuant (Xiao et al., 2023) migrate quantization difficulty from activations to weights, and LLM.int8() (Dettmers et al., 2022) demonstrates 8-bit inference at scale. For the KV cache specifically, KIVI (Liu et al., 2024) and KVQuant (Hooper et al., 2024) demonstrate 2- to 4-bit per-channel KV quantization that preserves quality on standard benchmarks; FP8 KV storage reduces the byte width when supported by a serving backend (Appendix L). These methods reduce per-element bytes without altering cache shape and compose multiplicatively with channel pruning, as discussed in §5.3. Per-channel quantization shares with our work the recognition that channels carry heterogeneous information and benefit from per-channel treatment; the two approaches are orthogonal in the storage axis they target (channel selection vs. bit precision per channel) and can be deployed together for additional savings.

**Attention sparsifiers and broader pruning.** A separate line of work sparsifies attention computation rather than the cache itself: Twilight (Lin et al., 2025) adaptively prunes attention via hierarchical top-$p$ thresholds, leaving the stored cache unchanged. These methods are conceptually orthogonal to the storage-reduction axis we address. Methodologically, GRIK draws on the broader pruning literature: the static norm-based prior follows Wanda (Sun et al., 2024) (originally for weight pruning) and related activation-aware criteria (Lin et al., 2024); SparseGPT (Frantar & Alistarh, 2023) extends one-shot weight pruning to LLM scale; movement pruning (Sanh et al., 2020) learns sparsity patterns adaptively via fine-tuning. The iterative refinement schedule connects to magnitude-based weight pruning (Han et al., 2015) and the lottery-ticket-style observation that progressively tightening masks recovers more accurate sparse subnetworks (Frankle & Carbin, 2019). We adapt these instruments to the KV-cache setting under structural GQA constraints.

## B Notation

Table 10 lists the symbols used throughout the paper.

## C Theoretical Analysis

This appendix formalizes the exact $k$-width shared-KV layout studied in the body. Section C.1 proves the K-cache footprint under C1 and C2. Section C.2 bounds the channel unions when either

Table 10: Notation used throughout the paper.

| Symbol | Meaning |
|---|---|
| $L$ | Number of transformer layers |
| $H$ | Number of query heads |
| $H_{\mathrm{kv}}$ | Number of key/value heads (KV heads) |
| $g = H/H_{\mathrm{kv}}$ | Group size: query heads per KV head |
| $d$ | Head dimension |
| $d_{\mathrm{model}}$ | Model hidden size, $d_{\mathrm{model}} = H \cdot d$ |
| $T$ | Sequence length |
| $B$ | Token-eviction budget (number of retained past tokens) |
| $w$ | Recent-window size, $w \leq B$ |
| $r \in [0, 1)$ | Channel pruning ratio (fraction dropped) |
| $k = d - \mathrm{round}(rd)$ | Number of retained channels per KV head |
| $\mathcal{G}_{h_{\mathrm{kv}}}$ | Set of query heads sharing KV head $h_{\mathrm{kv}}$ |
| $M_{h_{\mathrm{kv}}} \in \{0, 1\}^d$ | Channel mask for KV head $h_{\mathrm{kv}}$ |
| $W_k^{(h_{\mathrm{kv}})}$, $W_O^{(h)}$ | K-projection (per KV head), output projection (per Q head) |
| $\alpha_h$ | Output-projection influence of query head $h$ (Eq. 8) |
| $E_q[h, d']$, $E_k[h_{\mathrm{kv}}, d']$ | Per-channel query/key energy on the current prompt (Eq. 10) |
| $s_{h_{\mathrm{kv}}}[d']$ | Channel score in KV head $h_{\mathrm{kv}}$ (Eq. 2) |
| $N$ | Number of refinement steps ($N$=4 recommended; main tables $N$=32) |

condition is violated and separates these bounds from the measured native ThinK and SparK layouts. Section C.3 accounts for the recent-window correction to token/channel composition; Section C.4 analyzes the refinement correction.

Throughout, we use the notation of Appendix B: $L$ is the number of layers, $H_q$ and $H_{kv}$ are the numbers of query and KV heads, $g = H_q/H_{kv}$ is the GQA group size, $d$ is the per-head channel dimension, $T$ is the prefilled context length, $k = d - \mathrm{round}(rd)$ is the number of channels kept after pruning, and $b$ is the per-element byte count of the storage representation (e.g., $b = 2$ for bf16). The K cache occupies a single contiguous tensor per layer, and a deployed kernel is assumed to read from this tensor under the dense, row-major layout (NVIDIA, 2024).

### C.1 Storage Invariance under (C1, C2)

**Theorem C.1** (Storage Invariance). *Let $\{M_{h_{kv}}\}_{h_{kv} \in [H_{kv}]}$ be a channel mask family with $|M_{h_{kv}}| = k$ for all $h_{kv}$, and let the K cache be stored as a contiguous dense tensor of shape $H_{kv} \times T \times k$ per layer. If (C1) and (C2) of Definition 2.1 hold, then the total K-cache storage is exactly*

$$S_K^{(\mathrm{C1,C2})} = b \cdot L \cdot H_{kv} \cdot T \cdot k. \tag{3}$$

*Proof.* By (C2), the channel mask $M_{h_{kv}}$ is shared across all $T$ tokens, so the active channels of head $h_{kv}$ form a fixed subset of size $k$ that does not depend on the token index. The dense tensor for head $h_{kv}$ at layer $\ell$ is therefore $T \times k$ regardless of which $k$ channels are kept. By (C1), every Q-head $h \in \mathcal{G}_{h_{kv}}$ in the GQA group sees the same active channel set when reading from $M_{h_{kv}}$, so no separate per-query-head storage is needed. Summing across $H_{kv}$ heads and $L$ layers and multiplying by $b$ bytes per element gives Equation 3. The expression depends on $k$ but not on which $k$ channels are kept, so storage is invariant to the specific selection. □

*Remark* C.1. Theorem C.1 is the formal statement of the design target named in §4: any channel mask family satisfying (C1) and (C2) achieves the same storage reduction $S_K = bLH_{kv}T \cdot k = (k/d)$ times the Full KV baseline of $bLH_{kv}T \cdot d$, regardless of how $k$ channels are scored or selected. GRIK is therefore exactly storage-reducing by construction; quality of selection determines accuracy but not memory.

### C.2 Storage Degradation under Condition Violations

**Proposition C.1** (C1 Violation and Group Union)**.** *Assume each query-head mask in a KV group $\mathcal{G}_{h_{\mathrm{kv}}}$ has $k$ selected channels, and let $U_{h_{\mathrm{kv}}} := \bigcup_{h \in \mathcal{G}_{h_{\mathrm{kv}}}} M_h^{\mathrm{query}}$. Then*

$$k \;\le\; |U_{h_{\mathrm{kv}}}| \;\le\; \min(gk, d). \tag{4}$$

*If at least two masks in the group differ, $|U_{h_{\mathrm{kv}}}| \ge k+1$. A single fixed channel-index map for that group must therefore have width greater than $k$ to retain every requested channel. No general lower bound of width $d$ follows.*

*Proof.* Every size-$k$ mask is contained in the union, giving $|U_{h_{\mathrm{kv}}}| \ge k$; two different size-$k$ masks have a union of at least $k+1$ elements. The union bound gives $|U_{h_{\mathrm{kv}}}| \le \sum_{h \in \mathcal{G}_{h_{\mathrm{kv}}}} |M_h^{\mathrm{query}}| = gk$, while the channel universe has size $d$. A fixed index map must include the union to represent all requested channels. □

*Remark* C.2 (Numerical example: native ThinK on LLaMA-3-8B). At $r$=0.4 ($g$=4, $d$=128, $k$=77), group-inconsistent size-$k$ masks may require a union ranging from 78 to 128 channels; this does not force every implementation to use $d$ channels. In the native ThinK-on-GQA layout accounted for in Figure 1, K and V are materialized at query-head granularity, giving an asymptotic K-only ratio $gk/d \approx 2.41\times$ and total-KV ratio $g(k/d+1)/2 \approx 3.20\times$ relative to uncompressed GQA storage.

**Proposition C.2** (C2 Violation and Token Union)**.** *Assume each token mask for a KV head has $k$ selected channels and let $U_{h_{\mathrm{kv}}} := \bigcup_{t \in [T]} M_{h_{\mathrm{kv}}}^{(t)}$. Then*

$$k \;\le\; |U_{h_{\mathrm{kv}}}| \;\le\; \min(Tk, d). \tag{5}$$

*If two token masks differ, $|U_{h_{\mathrm{kv}}}| \ge k+1$, so one token-independent index map of width $k$ cannot retain every active channel. This does not imply that every implementation must store all $d$ channels.*

*Proof.* Every size-$k$ token support lies in $U_{h_{\mathrm{kv}}}$, and distinct size-$k$ supports have a union of at least $k+1$ elements. The union bound and the dimension of the channel universe give the upper bound. A fixed $k$-width map cannot contain a larger union. The evaluated SparK recovery implementation materializes a full-shaped $H_{\mathrm{kv}} \times T \times d$ K tensor, so its observed dense K footprint remains at the baseline; this behavior is specific to that implementation. □

### C.3 Composition with Token Eviction

**Proposition C.3** (Token and Channel Composition with a Recent Window)**.** *Suppose token eviction retains $B \le T$ positions, including $w \le B$ recent positions whose K vectors remain full-width. With one shared $k$-channel mask per KV head for the remaining $B-w$ positions, the K and V data storage is*

$$S_K = b_K L H_{kv}\big(wd + (B-w)k\big), \qquad S_V = b_V L H_{kv} B d. \tag{6}$$

*Relative to full-width K at the same token budget, the K-data factor is $w/B + (1-w/B)k/d$. Relative to full-width K at context length $T$, it is $(B/T)[w/B + (1-w/B)k/d]$. For $w=0$ this becomes the exact product $(B/T)(k/d)$; it approaches that product when $B \gg w$. These formulas omit mask metadata and allocator overhead.*

*Proof.* The $w$ recent K vectors each have $d$ entries per KV head, and the other $B-w$ each have $k$; V retains $d$ entries at all $B$ positions. Dividing the resulting data-byte counts by their respective full-width references gives the stated ratios. □

### C.4 Unit-Variance Correction during Iterative Refinement

**Proposition C.4** (Logit Variance Preservation)**.** *For a* fixed, input-independent *mask $M \subseteq [d]$, assume the pairs $(q_i, k_i)$ are independent across $i$, and $q_i$ and $k_i$ are independent within each pair, zero-mean, and unit-variance. The standard scaled dot-product attention (Vaswani et al., 2017) then produces a logit*

$$\ell_d \;=\; \frac{1}{\sqrt{d}} \sum_{i=1}^{d} q_i k_i \tag{7}$$

*with* $\mathrm{Var}(\ell_d) = 1$. *If a channel mask* $M \subset [d]$ *with* $|M| = k_t$ *is applied at refinement step* $t$, *the corresponding masked logit* $\ell_{k_t} := \frac{1}{\sqrt{k_t}} \sum_{i \in M} q_i k_i$ *also satisfies* $\mathrm{Var}(\ell_{k_t}) = 1$, *whereas using the unpruned scaling factor would give variance* $k_t/d$.

*Proof.* By independence and zero-mean, $\mathrm{Var}(q_i k_i) = \mathbb{E}[q_i^2 k_i^2] - \mathbb{E}[q_i k_i]^2 = \mathbb{E}[q_i^2]\mathbb{E}[k_i^2] - 0 = 1 \cdot 1 = 1$. The cross terms $\mathbb{E}[q_i k_i q_j k_j]$ for $i \neq j$ vanish under independence, so the sum $\sum_{i \in M} q_i k_i$ has variance $\sum_{i \in M} \mathrm{Var}(q_i k_i) = |M| = k_t$. Dividing by $\sqrt{k_t}$ yields unit variance. Using the unpruned scaling $1/\sqrt{d}$ instead gives variance $k_t/d$, which is below 1 for $k_t < d$ and would shift the softmax temperature implicitly used by the pre-trained model. □

*Remark* C.3. At deployment the attention implementation uses the model's original $1/\sqrt{d}$ scale. The calculation above concerns a fixed mask with independent, standardized random components; adaptive masks and learned transformer representations need not have unit logit variance. The $1/\sqrt{k_{t-1}}$ refinement scale is a heuristic for evaluating head influence, and does not modify the fixed prompt energies.

*Remark* C.4 (Independence assumption.). The independent, standardized coordinate model is only a motivating approximation for attention scaling. In trained transformers, coordinate dependence and selecting a mask from the prompt can invalidate its exact variance conclusion. The ThinK-mean comparison changes several score components simultaneously and does not isolate the effect of this scaling.

# D Method Details

This appendix provides the full definitions of the three signals that compose the channel score in Equation 2 (output-projection influence $\alpha_h$, static prior $w_k$, dynamic signal $E_q \cdot E_k$), together with the iterative-refinement procedure and the unit-variance correction referenced in §4.

## D.1 Output-Projection Influence

**Background.** Attention heads are known to contribute unequally to model output. Michel et al. (2019) demonstrated that a substantial fraction of heads in trained transformers can be removed at test time with little degradation, and Voita et al. (2019) showed that a small number of specialized heads carry functionally distinct roles. We reuse this insight not for head removal but as a criterion for weighting channel preferences within a GQA group.

**Definition.** Let $\mathbf{a}_h^{(t)}(u)$ be the attention output of query head $h$ at observed query $u$ under mask $M^{(t-1)}$. With $W_O^{(h)}$ the output-projection block for this head, define

$$\phi_h^{(t)} = \frac{1}{|\mathcal{O}_t|} \sum_{u \in \mathcal{O}_t} \|W_O^{(h)} \mathbf{a}_h^{(t)}(u)\|_2, \qquad \alpha_h^{(t)} = \frac{\max(\phi_h^{(t)}, 0)}{\max(\sum_{h' \in \mathcal{G}_j} \max(\phi_{h'}^{(t)}, 0), 10^{-12})}, \quad h \in \mathcal{G}_j. \tag{8}$$

Here $\mathcal{O}_t$ is the step's portion of a 32-query observation window. This group normalization weights heads by their measured contribution to the residual stream; the weights change when attention is recomputed under a new mask.

## D.2 Static Prior and Dynamic Signal

GRIK defines channel importance as the product of a weight-level static prior and a prompt-specific dynamic signal. This product structure extends the principle established in LLM compression research—that channel importance scales with the combination of weight magnitude and activation magnitude (Sun et al., 2024; Lin et al., 2024)—to the GQA channel-selection setting.

**Static prior.** Let $\widetilde{w}_k[j, c] = \|W_k^{(j)}[c, :]\|_2$ be the K-projection row norm for output channel $c$ of KV head $j$. We normalize by its within-head mean:

$$w_k[j, c] = \frac{\widetilde{w}_k[j, c]}{d^{-1} \sum_{c'=1}^{d} \widetilde{w}_k[j, c']}. \tag{9}$$

**Algorithm 1** GRIK channel mask computation

**Require:** Prefilled $\mathbf{Q}, \mathbf{K}, \mathbf{V}$; pruning ratio $r$; steps $N$; output-projection blocks $W_O^{(h)}$
**Ensure:** Group-shared masks $\{M_j\}_{j\in[H_{\mathrm{kv}}]}$

```
Compute w_k by Equation 9; compute E_q, E_k once by Equation 10
Initialize M_j^(0) ← 1_d; set k_0 ← d and k ← d − round(rd)
Split 32 observation queries into N step groups O_t
for t = 1, ..., N do
    Compute attention outputs on O_t under M_j^(t−1) with 1/√k_{t−1} scaling
    Compute α_h^(t) by Equation 8
    k_t ← scheduled width decreasing to k at step N
    s_j^(t)[c] ← w_k[j, c] E_k[j, c] Σ_{h∈G_j} α_h^(t) E_q[h, c] for every c ∈ [d]
    M_j^(t) ← TopK(s_j^(t), k_t) for every j
end for
return {M_j^(N)}; decode with 1/√d scaling
```

This prompt-independent term is computed once per model checkpoint.

**Dynamic signal.** We aggregate per-channel energy on the current prompt separately on the query and key sides:

$$E_q[h, d'] = \mathbb{E}_t\left[q^2_{h,t,d'}\right], \qquad E_k[h_{\mathrm{kv}}, d'] = \mathbb{E}_t\left[k^2_{h_{\mathrm{kv}},t,d'}\right]. \tag{10}$$

The product $E_q \cdot E_k$ reflects the contribution of each channel to the attention dot product on the current prompt. Related attention-based dynamic importance signals have been used for token-level KV compression (Zhang et al., 2023; Li et al., 2024; Xu et al., 2025); we apply the same principle along the channel axis.

### D.3 Iterative Refinement

**Fixed energies and changing head weights.** The squared query/key energies in Equation 10 are computed once on the prompt and remain fixed. Attention under a changing mask alters the observed outputs and hence $\alpha_h^{(t)}$, which changes the group score in Equation 2. The 32 most recent observation queries are split into $N$ groups (32, 8, or 1 query per step for $N$=1, 4, 32). At each step, scores are evaluated for all $d$ channels and the top $k_t$ are selected afresh; a previously dropped channel can return.

**Unit-variance scaling.** When the current mask has $k_{t-1}$ active channels, refinement attention divides their dot product by $\sqrt{k_{t-1}}$. Deployment attention instead uses the original $\sqrt{d}$ scale. This is a heuristic guided by the independent unit-variance calculation in Appendix C.4; the actual mask and transformer features do not satisfy that calculation's independence assumptions.

### D.4 Pseudocode

The complete GRIK channel-mask computation is summarized in Algorithm 1.

## E Experimental Setup

This appendix gives the full setup for the experiments summarized in §5.1: per-backbone architectural specifications (§E.1), benchmark protocols and metrics (§E.2), baseline implementation details (§E.3), and hardware/runtime configurations (§E.4).

### E.1 Base Models

We evaluate GRIK across three GQA-based instruction-tuned LLMs spanning two model families and three GQA configurations. Architectural specifications are summarized in Table 11.

Table 11: Backbone architectures used in our experiments. $L$ = transformer layers; $H_q$, $H_{kv}$ = query and KV heads; $g = H_q/H_{kv}$ = GQA group size; $d$ = per-head channel dimension; $d_{\text{model}}$ = hidden size; $d_{\text{ffn}}$ = feed-forward inner dimension; $V$ = vocabulary size.

| **Model** | $L$ | $H_q$ | $H_{kv}$ | $g$ | $d$ | $d_{\text{model}}$ | $d_{\text{ffn}}$ | $V$ | **max ctx** |
|---|---|---|---|---|---|---|---|---|---|
| LLaMA-3-8B-Instruct | 32 | 32 | 8 | 4 | 128 | 4096 | 14336 | 128256 | 8K |
| LLaMA-3.1-8B-Instruct | 32 | 32 | 8 | 4 | 128 | 4096 | 14336 | 128256 | 128K |
| Qwen-2.5-7B-Instruct | 28 | 28 | 4 | 7 | 128 | 3584 | 18944 | 152064 | 128K |

**LLaMA-3-8B-Instruct.** Native context length 8K, no RoPE scaling. We use the released weights at the standard precision (bfloat16). The relatively small group factor $g = 4$ makes the GQA-induced storage savings of GRIK proportionally moderate ($k/d \cdot 1/g$ of MHA storage at the same $r$), but the structural pattern is identical to the higher-$g$ models.

**LLaMA-3.1-8B-Instruct.** Same architecture as LLaMA-3-8B-Instruct except for RoPE scaling: a frequency factor of 8 extends the native 8K context to 128K. We load the model with `transformers` 4.45.2 to enable the `rope_scaling` parameter; the `unsloth/Meta-Llama-3.1-8B-Instruct` mirror is used because the gated official Meta release requires special access (the weights are bit-identical). RULER experiments require the long-context capability of this backbone.

**Qwen-2.5-7B-Instruct.** 28 layers, 28 query heads, 4 KV heads, hidden size 3584. The larger group factor $g = 7$ amplifies both the storage gains under (C1, C2)-conformant pruning and the failure modes for (C1)/(C2) violations. We load the model in bfloat16 on an A100 80 GB.

### E.2 Datasets and Metrics

We evaluate on two long-context benchmarks chosen to stress complementary axes of model capability.

**LongBench-v1 (Bai et al., 2024).** A multi-task suite of 16 datasets aggregated to a single 16-task average score (LongBench-AVG). Tasks span six categories:

- **Single-document QA**: NarrativeQA, Qasper, MultiFieldQA-en (F1).
- **Multi-document QA**: HotpotQA, 2WikiMQA, MuSiQue (F1).
- **Summarization**: GovReport, QMSum, MultiNews (ROUGE-L).
- **Few-shot learning**: TREC (classification accuracy), TriviaQA (F1), SAMSum (ROUGE-L).
- **Synthetic**: PassageCount, PassageRetrieval-en (accuracy).
- **Code completion**: LCC, RepoBench-P (Edit-Sim).

Inputs range from 4K to 32K tokens; metric per task follows the original LongBench specification. Per-task numbers in our paper are computed with the same scoring scripts as the official benchmark.

**RULER (Hsieh et al., 2024).** A controllable synthetic long-context benchmark spanning 13 subtasks: needle-in-a-haystack variants (`niah_single_1/2/3`, `niah_multikey_1/2/3`, `niah_multivalue`, `niah_multiquery`), variable tracking (`vt`), common-word extraction (`cwe`), frequent-word extraction (`fwe`), and two question-answering tasks (`qa_1`, `qa_2`). We evaluate at sequence lengths $\{4\text{K}, 8\text{K}, 16\text{K}, 32\text{K}\}$ with 200 prompts per (task, length) cell, in the no-eviction regime (KV token budget set to 200K so that H2O eviction never fires) at pruning ratios $r{=}0.4$ (primary setting) and $r{=}0.7$ (aggressive stress test). RULER's needle subtasks measure *exact-string retrieval*; comparisons across these implementations can probe the effect of token-dependent recovery, although they do not isolate condition C2 from scoring and recovery differences. We use the official RULER prediction harness with one minor patch: the `evaluate.py` script's heavy `nemo` dependency is replaced by a minimal manifest reader (full patch in §E.4).

### E.3 BASELINES

We compare GRIK against the unpruned cache, two channel-pruning baselines, and one storage-controlled variant. All baselines share the same H2O (Zhang et al., 2023) token-eviction prefix at the same KV token budget so that quality differences reflect channel-level effects only. Method-specific implementation details follow.

**Full KV.** The native GQA cache without channel pruning, paired with H2O eviction at budget $B$. Serves as the reference against which channel-pruning methods are measured at matched memory budget. For LLaMA-3-8B at $r$=0.4, Full KV scores are cited from Xu et al. (2025) Table 2 (we use the same evaluation harness, prompts, and decoding configuration); for LLaMA-3.1 and Qwen-2.5 we measure Full KV ourselves under the same configuration as the channel-pruning methods to ensure consistency.

**ThinK (Xu et al., 2025).** We use the authors' public MHA-style implementation with per-query-head attention-magnitude scoring (`thunlp/ThinK`, commit `4cf166a`). Its native GQA application uses query-head granularity for the measured K and V cache, yielding approximately $2.41\times$ K-only and $3.20\times$ total K+V storage at $r$=0.4 on LLaMA-3-8B (§5.3). These implementation-specific ratios are not universal lower bounds for arbitrary masks that violate C1. LLaMA-3-8B ThinK quality numbers are cited from Xu et al. (2025) Table 2; we run the released code for LLaMA-3.1 and Qwen-2.5.

**ThinK-mean.** A storage-controlled variant of ThinK introduced in §3: the per-query-head ThinK score is averaged uniformly within each KV group, producing a single shared mask per KV head. This variant matches GRIK's storage footprint exactly ($H_{kv}$ heads, $k$ channels) and isolates the contribution of GRIK's $\alpha_h$-weighted aggregation, static prior, and iterative refinement from the underlying storage layout. Used only for the score-form ablation (Appendix J); not part of the main result table.

**SparK (Liao et al., 2026).** Query-aware unstructured channel sparsity. Each token receives an independent per-channel mask; pruned entries are not packed into a reduced tensor but reconstructed in place by drawing from a per-channel distribution at the recovered position. We use the released implementation with the default `normal` recovery mode (recovery distribution: $\mathcal{N}(\mu_c, \sigma_c^2)$ where $\mu_c, \sigma_c$ are estimated per channel from the unmasked statistics). Because masks vary across tokens, condition (C2) is violated; the recovered cache has shape $H_{kv} \times T \times d$, identical to Full KV's storage footprint (Proposition C.2).

**GRIK (Ours).** The method described in §4 uses $\alpha_h$-weighted aggregation, a Wanda-style $\|W_k\|$ static prior (Sun et al., 2024), and refinement scaling motivated by Proposition C.4. We recommend four refinement steps for practical use; the main benchmark tables use 32 steps, and the four-step comparison is reported in §5.4. The schedule for $k_t$ at intermediate refinement steps follows a linear schedule from $d$ to $d - \mathrm{round}(rd)$.

**Common settings.** All methods use identical greedy decoding (temperature 0). The LongBench budget sweep uses the same H2O (Zhang et al., 2023) eviction prefix with attention-score accumulation, a recent window $w = 32$, pruning ratios $r \in \{0.4, 0.6\}$, and token budgets $B \in \{128, 256, 512, 1024, 2048\}$ plus a no-eviction control. The recent $w$ tokens are stored at the original (unpruned) channel dimension $d$, following the ThinK and H2O protocols. RULER instead uses a 200K budget so eviction does not activate, with $r$=0.4 as the primary ratio and $r$=0.7 as an aggressive stress test.

### E.4 HARDWARE AND RUNTIME

**Hardware.** LLaMA-3 experiments run on a single NVIDIA RTX A6000 (48 GB) box. LLaMA-3.1 LongBench experiments run on a DGX H100 cluster (single GPU per run). Qwen-2.5 experiments run on a single A100 80 GB. RULER 4K–16K runs on the A6000 box; the 32K RULER cell runs on an RTX PRO 6000 (96 GB) because peak prefill memory at 32K exceeds the A6000 envelope. The

Table 12: **Measured KV-cache memory.** Materialized K+V+mask storage (MiB) on LLaMA-3-8B-Instruct.

| **Method** | $B$=128 | $B$=512 | $B$=1024 | $B$=2048 |
|---|---|---|---|---|
| $r = 0.4$ | | | | |
| Full KV | 16.00 | 64.00 | 128.00 | 256.00 |
| ThinK | 54.56 | 208.31 | 413.31 | 823.31 |
| SparK | 16.00 | 64.00 | 128.00 | 256.00 |
| **GRIK** | **13.64** | **52.08** | **103.33** | **205.83** |
| $r = 0.6$ | | | | |
| Full KV | 16.00 | 64.00 | 128.00 | 256.00 |
| ThinK | 49.69 | 183.94 | 362.94 | 720.94 |
| SparK | 16.00 | 64.00 | 128.00 | 256.00 |
| **GRIK** | **12.42** | **45.98** | **90.73** | **180.23** |

TRT-LLM KV-pool validation runs on a Hopper-class GPU using the official TensorRT-LLM 1.0.0 docker image.

**Runtime configuration.** `transformers` 4.40 for LLaMA-3 and Qwen-2.5; `transformers` 4.45.2 for LLaMA-3.1 (required for `rope_scaling`). All forward passes use eager attention (the GRIK and ThinK monkey-patches operate on the eager path); for prefill at $\geq$ 8K tokens we additionally enable `prefill_sdpa=1` which routes the prefill output projection through SDPA flash-attention while keeping cache writes on the eager path (RULER setup; see `docs/experiments/main_table/llama3.1-8b-instruct/ruler/RUNNING.md` in the released code).

**RULER patch.** The official RULER `evaluate.py` imports `nemo.collections.asr.parts.utils.manifest_utils`, which pulls in the entire NeMo dependency tree (incompatible with our environment). We replace the import with a one-line manifest reader/writer using the standard `json` module; the patched function preserves bit-equivalent behavior on RULER's manifest format. The patch is included in our released code.

**Reproducibility.** The evaluation harness, prompts, and decoding configuration are shared across all methods within a given backbone. Seeds are fixed at the dataset-loader level (LongBench/RULER both ship with deterministic prompt orderings). With the released code and the runtime configuration above, all numbers in this paper reproduce within the floating-point determinism of the underlying CUDA kernels (typically $\leq 0.05$ on a 16-task LongBench AVG).

# F Memory Validation and Detailed Memory Grid

## F.1 Formula vs. Realized Cache

The primary grid in Table 12 reports materialized total K+V tensor bytes, including a small mask overhead. A consistency calculation for LLaMA-3-8B-Instruct ($L$=32, $H_{kv}$=8, $d$=128, bf16, $w$=32) gives 16.000 and 128.000 MiB for full K+V at $B$=128 and 1024. For GRIK at $r$=0.4, $k$=77 and Equation C.3 predict 13.609 and 103.297 MiB; the main measured grid gives 13.64 and 103.33 MiB. The approximately 0.03 MiB difference is consistent with additional mask storage. This calculation checks the main grid against data-byte accounting; it is not an independent serving allocator measurement. For $r$=0.6, $B$=128, the predicted data footprint is 12.391 MiB ($k$=51), consistent with the measured 12.42 MiB. Earlier inconsistent validation entries are omitted.

# G Prefill Latency Measurement

We measure end-to-end prefill wall-clock time on LLaMA-3-8B-Instruct at $r$=0.4 ($d_{\text{keep}}$=77), batch 1, bf16, on an NVIDIA GB10 GPU, across context lengths $\{4\text{K}, 8\text{K}, 16\text{K}, 32\text{K}\}$. Three configurations are compared: Full KV prefill (baseline), GRIK with $N$=1 refinement step (one-shot scoring), and

Table 13: Prefill wall-clock latency on LLaMA-3-8B-Instruct, $r$=0.4, NVIDIA GB10, bf16, batch 1. Overheads are relative to Full KV prefill at the same context length. The note below the table reports E2E throughput at 16 generated tokens vs. Full KV.

| | Full KV | | GRIK $N$=1 | | GRIK $N$=32 | |
|---|---|---|---|---|---|---|
| **Context** | prefill (s) | E2E (rel.) | prefill (s) | overhead | prefill (s) | overhead |
| 4K | 1.20 | 1.00× | 1.37 | +14.0% | 2.97 | +147.7% |
| 8K | 2.46 | 1.00× | 2.76 | +11.9% | 5.12 | +108.1% |
| 16K | 5.19 | 1.00× | 5.72 | +10.1% | 8.98 | +73.0% |
| 32K | 12.05 | 1.00× | 13.03 | +8.2% | 18.45 | +53.1% |

E2E (rel.) at 16 generated tokens: GRIK $N$=1 achieves 0.96–0.98×; GRIK $N$=32 achieves 0.62–0.74× Full KV.

GRIK with the $N$=32 configuration used in the main benchmark tables. The practical $N$=4 prefill time was not measured in this experiment. Each cell is the mean over 3 runs after 1 warmup, with standard deviations below 1% of the mean. The note below the table gives the range of relative end-to-end (E2E) throughput at 16 generated tokens.

**Two patterns are visible.** (i) The $N$=32 relative overhead falls monotonically with context length, from +147.7% at 4K to +53.1% at 32K. This is expected: the additional refinement work is spread over these measured context lengths. Each step also recomputes attention on observation queries, so its total cost depends on $T$; the measurements do not establish an asymptotic vanishing overhead. (ii) One-shot ($N$=1) adds 8.2–14.0% prefill overhead over the measured 4K–32K contexts. The remaining gap between $N$=1 and $N$=32 is the additional refinement steps and their attention recomputation; the measured 32K difference between $N$=1 and $N$=32 is 5.42 s in total.

**End-to-end and deployment trade-off.** At 16 generated tokens (note below the table), GRIK $N$=32 runs at 0.62–0.74× Full KV throughput across the context range. For deployments with long generations, the one-time prefill overhead amortizes; for short-generation serving, the latency–memory trade-off depends on whether the workload is bound by latency or by KV-cache memory pressure. Refinement steps in our implementation are not yet batched across layers; per-layer batching would further reduce the $N$=32 overhead.

# H LongBench Detailed Results

This appendix reports per-task LongBench-v1 scores that complement the summary in Table 2, grouped by backbone.

## H.1 LLaMA-3-8B-Instruct

Table 14 reports per-task scores at both $r$=0.4 and $r$=0.6. Full KV at $r$=0.4 values are taken from Xu et al. (2025) Table 2 (LLaMA-3-8B-Instruct, H2O baseline at matched KV size). The 8h GQA-aggregated ThinK variant (ThinK-mean) is reported separately in Appendix J as a score-form ablation.

## H.2 LLaMA-3.1-8B-Instruct

Table 15 reports per-task scores at both pruning ratios, all five KV budgets, and the no-eviction (channel-only) ceiling. All entries are our measurements; the $r$=0.4 sub-block matches the operating point used for the diagnostic in §3 (Table 1b) and the trajectory view in Figure 2. The per-task pattern aligns with the 16-task aggregate in Table 2: GRIK matches or exceeds ThinK and SparK on the majority of tasks at every budget, and the gap between GRIK and Full KV stays below 0.27 points at every budget at $r$=0.4. The largest per-task differentiation appears in retrieval-style subtasks (NarrativeQA, Qasper, HotpotQA, PassageRetrieval), consistent with the C2 dissociation tested more directly on RULER (§5.5). At $r$=0.6, SparK's degradation is monotone in $B$ and concentrates on the same retrieval-heavy tasks; ThinK and GRIK degrade gracefully across all categories.

Table 14: Per-task LongBench on LLaMA-3-8B-Instruct, $w$=32, both pruning ratios. **Bold**: best per-block Avg among ThinK / SparK / GRIK; "–": not available. Full KV and ThinK rows at $r$=0.4, $B \in \{128, 512, 1024, 2048\}$, and ThinK rows at $r$=0.6, $B \in \{512, 1024, 2048\}$, cited from Xu et al. (2025) Table 2.

| Method | Single-Doc QA | | | Multi-Doc QA | | | Summarization | | | Few-shot | | | Synth. | | Code | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NrtvQA | Qasper | MF-en | HotpotQA | 2WikiMQA | Musique | GovReport | QMSum | MultiNews | TREC | TriviaQA | SAMSum | PCount | PRe | Lcc | RB-P | Avg. |
| ***Channel pruning*** $r = 0.4$ ***(60% kept)*** | | | | | | | | | | | | | | | | | |
| $B$=128 | | | | | | | | | | | | | | | | | |
| Full KV | 22.12 | 13.20 | 31.61 | 37.79 | 32.71 | 18.45 | 20.32 | 22.02 | 21.10 | 38.50 | 87.75 | 39.14 | 5.83 | 69.50 | 55.06 | 50.97 | 35.38 |
| ThinK (Xu et al., 2025) | 22.85 | 14.55 | 29.49 | 38.63 | 30.84 | 18.90 | 20.12 | 21.96 | 20.68 | 38.50 | 86.38 | 38.40 | 5.50 | 69.17 | 57.93 | 56.12 | 35.63 |
| SparK (Liao et al., 2026) | 20.30 | 8.74 | 23.46 | 31.51 | 28.13 | 14.42 | 19.18 | 21.23 | 20.20 | 40.50 | 89.55 | 37.13 | 6.03 | 66.48 | 50.82 | 46.02 | 32.73 |
| **GRIK** (Ours) | 21.64 | 12.31 | 31.87 | 40.69 | 31.24 | 19.31 | 19.40 | 21.75 | 19.79 | 40.50 | 89.78 | 38.98 | 5.55 | 69.50 | 57.75 | 54.72 | **35.92** |
| $B$=512 | | | | | | | | | | | | | | | | | |
| Full KV | 23.52 | 17.93 | 34.68 | 42.11 | 33.52 | 19.92 | 22.11 | 22.56 | 23.82 | 41.00 | 90.46 | 40.20 | 5.87 | 69.50 | 56.71 | 51.69 | 37.23 |
| ThinK (Xu et al., 2025) | 23.76 | 17.80 | 33.80 | 40.39 | 30.70 | 19.09 | 21.82 | 22.51 | 23.78 | 41.00 | 90.16 | 40.67 | 5.15 | 69.25 | 60.77 | 57.58 | 37.39 |
| SparK (Liao et al., 2026) | 21.79 | 10.21 | 27.28 | 29.81 | 26.59 | 14.98 | 21.09 | 22.79 | 23.08 | 53.50 | 90.80 | 37.78 | 6.04 | 65.26 | 47.45 | 44.19 | 33.92 |
| **GRIK** (Ours) | 23.50 | 18.76 | 37.54 | 42.49 | 29.59 | 20.42 | 21.94 | 22.57 | 23.50 | 52.50 | 90.36 | 39.76 | 5.32 | 69.33 | 61.82 | 58.76 | **38.63** |
| $B$=1024 | | | | | | | | | | | | | | | | | |
| Full KV | 25.62 | 22.16 | 36.81 | 41.01 | 33.53 | 19.41 | 23.28 | 22.65 | 25.41 | 46.50 | 90.82 | 41.78 | 5.79 | 69.25 | 59.69 | 55.50 | 38.70 |
| ThinK (Xu et al., 2025) | 25.52 | 21.93 | 37.17 | 41.56 | 31.22 | 20.17 | 22.89 | 22.95 | 25.21 | 47.00 | 90.74 | 41.34 | 5.57 | 69.50 | 62.58 | 58.67 | 39.00 |
| SparK (Liao et al., 2026) | 22.43 | 14.20 | 28.20 | 30.63 | 25.38 | 16.01 | 21.40 | 23.23 | 24.74 | 62.00 | 89.63 | 38.86 | 5.01 | 64.32 | 45.53 | 43.24 | 34.68 |
| **GRIK** (Ours) | 24.77 | 22.93 | 38.49 | 42.82 | 31.95 | 20.40 | 23.32 | 22.61 | 25.33 | 63.00 | 90.36 | 40.11 | 5.30 | 68.96 | 61.83 | 59.80 | **40.12** |
| $B$=2048 | | | | | | | | | | | | | | | | | |
| Full KV | 25.56 | 26.85 | 39.54 | 44.30 | 32.92 | 21.09 | 24.68 | 23.01 | 26.16 | 53.00 | 90.65 | 41.84 | 4.91 | 69.25 | 58.43 | 51.31 | 39.59 |
| ThinK (Xu et al., 2025) | 25.56 | 26.31 | 39.20 | 42.96 | 31.81 | 20.53 | 24.23 | 23.35 | 25.90 | 53.50 | 90.56 | 41.03 | 5.52 | 69.25 | 62.10 | 59.00 | 40.05 |
| SparK (Liao et al., 2026) | 24.51 | 13.87 | 27.46 | 30.45 | 24.95 | 11.90 | 21.50 | 23.15 | 25.36 | 68.00 | 89.44 | 38.14 | 4.13 | 65.46 | 46.57 | 42.40 | 34.83 |
| **GRIK** (Ours) | 25.46 | 28.05 | 38.80 | 43.06 | 31.18 | 21.06 | 25.14 | 23.03 | 26.26 | 71.00 | 90.42 | 40.97 | 4.73 | 69.17 | 62.66 | 59.87 | **41.30** |
| ***Channel pruning*** $r = 0.6$ ***(40% kept — aggressive pruning)*** | | | | | | | | | | | | | | | | | |
| $B$=128 | | | | | | | | | | | | | | | | | |
| Full KV | 22.12 | 13.20 | 31.61 | 37.79 | 32.71 | 18.45 | 20.32 | 22.02 | 21.10 | 38.50 | 87.75 | 39.14 | 5.83 | 69.50 | 55.06 | 50.97 | 35.38 |
| ThinK (Xu et al., 2025) | 22.93 | 11.13 | 30.43 | 38.46 | 30.18 | 17.19 | 19.45 | 22.01 | 19.86 | 39.00 | 86.73 | 37.80 | 5.47 | 69.00 | 56.06 | 55.30 | **35.06** |
| SparK (Liao et al., 2026) | 17.34 | 5.58 | 17.76 | 18.71 | 16.07 | 9.00 | 17.37 | 21.43 | 18.53 | 41.00 | 86.36 | 35.16 | 5.83 | 57.43 | 40.95 | 38.95 | 27.97 |
| **GRIK** (Ours) | 21.79 | 11.74 | 31.04 | 39.55 | 28.83 | 19.28 | 18.48 | 21.72 | 18.88 | 39.00 | 86.92 | 37.90 | 5.60 | 69.50 | 55.09 | 54.65 | 35.00 |
| $B$=512 | | | | | | | | | | | | | | | | | |
| Full KV | 23.52 | 17.93 | 34.68 | 42.11 | 33.52 | 19.92 | 22.11 | 22.56 | 23.82 | 41.00 | 90.46 | 40.20 | 5.87 | 69.50 | 56.71 | 51.69 | 37.23 |
| ThinK (Xu et al., 2025) | 23.40 | 14.83 | 32.62 | 38.47 | 30.97 | 19.81 | 20.80 | 22.04 | 21.60 | 40.00 | 88.79 | 38.90 | 5.36 | 69.50 | 58.28 | 57.65 | 36.44 |
| SparK (Liao et al., 2026) | 16.33 | 6.36 | 16.36 | 16.17 | 15.69 | 6.98 | 14.76 | 20.06 | 16.72 | 51.00 | 84.77 | 31.77 | 5.22 | 52.39 | 36.14 | 35.17 | 26.62 |
| **GRIK** (Ours) | 23.21 | 15.30 | 37.30 | 41.09 | 33.80 | 20.84 | 20.65 | 21.89 | 21.58 | 47.50 | 88.55 | 37.18 | 5.84 | 69.17 | 58.24 | 57.13 | **37.45** |
| $B$=1024 | | | | | | | | | | | | | | | | | |
| Full KV | 25.62 | 22.16 | 36.81 | 41.01 | 33.53 | 19.41 | 23.28 | 22.65 | 25.41 | 46.50 | 90.82 | 41.78 | 5.79 | 69.25 | 59.69 | 55.50 | 38.70 |
| ThinK (Xu et al., 2025) | 24.06 | 17.80 | 37.85 | 38.63 | 29.98 | 19.40 | 21.41 | 22.32 | 23.42 | 44.50 | 90.16 | 39.43 | 5.84 | 69.50 | 58.31 | 58.73 | 37.58 |
| SparK (Liao et al., 2026) | 13.98 | 6.59 | 12.90 | 13.13 | 10.65 | 4.77 | 12.66 | 18.61 | 14.65 | 52.75 | 77.86 | 29.06 | 4.56 | 47.98 | 33.76 | 31.83 | 24.11 |
| **GRIK** (Ours) | 25.07 | 18.73 | 39.29 | 40.43 | 32.45 | 20.08 | 20.88 | 22.57 | 22.92 | 54.50 | 88.77 | 37.63 | 5.09 | 69.00 | 59.05 | 58.69 | **38.45** |
| $B$=2048 | | | | | | | | | | | | | | | | | |
| Full KV | 25.56 | 26.85 | 39.54 | 44.30 | 32.92 | 21.09 | 24.68 | 23.01 | 26.16 | 53.00 | 90.65 | 41.84 | 4.91 | 69.25 | 58.43 | 51.31 | 39.59 |
| ThinK (Xu et al., 2025) | 24.37 | 22.14 | 37.77 | 40.13 | 29.50 | 20.26 | 22.09 | 22.76 | 24.78 | 49.50 | 90.16 | 39.69 | 5.56 | 69.50 | 59.24 | 58.78 | 38.51 |
| SparK (Liao et al., 2026) | 11.46 | 6.15 | 12.89 | 12.31 | 9.25 | 4.40 | 11.70 | 16.19 | 13.25 | 50.00 | 71.76 | 24.59 | 4.54 | 41.21 | 30.83 | 30.21 | 21.92 |
| **GRIK** (Ours) | 24.52 | 21.82 | 39.71 | 40.59 | 30.41 | 19.29 | 22.12 | 23.11 | 23.90 | 62.50 | 89.69 | 37.93 | 5.81 | 68.29 | 58.87 | 59.26 | **39.24** |

### H.3 QWEN2.5-7B-INSTRUCT

Table 16 reports per-task scores on Qwen2.5-7B-Instruct, ($g$=7) backbone. The Full KV row in the no-eviction sub-block is $r$-independent and is repeated in both ratio panels for readability. Two patterns dominate the per-task view. First, SparK's collapse to AVG 11 noted in §5.2 is not driven by a small number of bad tasks: the per-token recovery operator hurts *every* task category, with most tasks scoring below 5. This is the per-task signature of the C2 violation amplified by the larger group factor (Appendix C.2). Second, the small ThinK–GRIK gap at $B$=128 on this backbone is concentrated on a handful of summarization and code-completion tasks where ThinK's per-query-head mask flexibility carries a residual benefit; from $B$=512 onward the differences attenuate and GRIK leads

Table 15: Per-task LongBench on LLaMA-3.1-8B-Instruct, $w$=32, both pruning ratios. **Bold**: best per-block Avg; "–": not available.

| Method | Single-Doc QA | | | Multi-Doc QA | | | Summarization | | | Few-shot | | | Synth. | | Code | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NrtvQA | Qasper | MF-en | HotpotQA | 2WikiMQA | Musique | GovReport | QMSum | MultiNews | TREC | TriviaQA | SAMSum | PCount | PRe | Lcc | RB-P | Avg. |
| ***Channel pruning*** $r = 0.4$ ***(60% kept)*** | | | | | | | | | | | | | | | | | |
| $B$=128 | | | | | | | | | | | | | | | | | |
| Full KV | 22.18 | 23.90 | 47.77 | 46.11 | 45.99 | 23.34 | 21.07 | 22.56 | 20.00 | 40.00 | 90.82 | 40.60 | 10.87 | 72.00 | 59.43 | 51.21 | 39.87 |
| ThinK (Xu et al., 2025) | 20.63 | 24.97 | 41.17 | 43.47 | 47.42 | 22.29 | 22.17 | 22.21 | 20.05 | 39.50 | 90.81 | 40.73 | 10.67 | 72.50 | 58.26 | 50.66 | 39.22 |
| SparK (Liao et al., 2026) | 20.16 | 23.24 | 45.07 | 45.11 | 43.12 | 20.51 | 20.61 | 22.21 | 20.12 | 42.00 | 91.04 | 39.46 | 10.40 | 68.67 | 54.11 | 45.72 | 38.22 |
| **GRIK** (Ours) | 21.27 | 23.72 | 47.42 | 45.43 | 46.10 | 23.92 | 21.10 | 22.47 | 19.46 | 40.50 | 90.20 | 40.46 | 10.87 | 72.00 | 58.32 | 50.31 | **39.60** |
| $B$=512 | | | | | | | | | | | | | | | | | |
| Full KV | 24.91 | 34.59 | 53.27 | 46.71 | 47.55 | 24.48 | 24.72 | 22.97 | 23.90 | 53.00 | 91.93 | 43.09 | 10.67 | 72.50 | 64.04 | 55.76 | 43.38 |
| ThinK (Xu et al., 2025) | 21.71 | 30.38 | 48.37 | 46.36 | 48.07 | 23.78 | 25.19 | 22.60 | 23.22 | 43.00 | 91.77 | 42.00 | 10.37 | 72.50 | 62.40 | 53.28 | 41.56 |
| SparK (Liao et al., 2026) | 23.89 | 33.81 | 49.03 | 43.65 | 42.70 | 19.03 | 22.72 | 22.77 | 23.60 | 48.50 | 91.32 | 38.81 | 10.18 | 70.17 | 54.16 | 44.68 | 39.94 |
| **GRIK** (Ours) | 25.34 | 34.36 | 53.44 | 47.36 | 47.31 | 24.70 | 24.27 | 22.92 | 23.31 | 52.00 | 91.98 | 41.74 | 10.83 | 72.50 | 63.46 | 55.19 | **43.17** |
| $B$=1024 | | | | | | | | | | | | | | | | | |
| Full KV | 25.52 | 41.00 | 55.62 | 47.96 | 48.25 | 24.64 | 26.88 | 23.36 | 25.74 | 60.50 | 91.93 | 42.53 | 10.38 | 72.50 | 64.82 | 56.38 | 44.88 |
| ThinK (Xu et al., 2025) | 23.66 | 36.53 | 53.03 | 47.27 | 46.95 | 24.27 | 26.56 | 22.65 | 25.45 | 45.00 | 92.44 | 41.84 | 6.99 | 72.20 | 64.41 | 55.40 | 42.79 |
| SparK (Liao et al., 2026) | 22.79 | 34.99 | 49.82 | 45.27 | 41.42 | 20.91 | 21.58 | 22.50 | 23.97 | 53.50 | 90.66 | 37.68 | 10.84 | 70.50 | 49.85 | 42.71 | 39.94 |
| **GRIK** (Ours) | 24.80 | 39.87 | 55.06 | 48.14 | 48.71 | 24.81 | 26.90 | 22.95 | 25.24 | 59.50 | 92.02 | 41.80 | 10.78 | 72.50 | 64.19 | 56.74 | **44.63** |
| $B$=2048 | | | | | | | | | | | | | | | | | |
| Full KV | 25.71 | 43.82 | 55.95 | 48.46 | 48.32 | 24.44 | 29.63 | 23.67 | 26.63 | 66.00 | 91.37 | 43.20 | 9.93 | 72.50 | 65.08 | 57.00 | 45.73 |
| ThinK (Xu et al., 2025) | 24.68 | 40.99 | 55.02 | 47.95 | 47.62 | 25.16 | 29.02 | 23.22 | 26.88 | 49.50 | 91.81 | 43.49 | 10.58 | 72.50 | 64.60 | 55.74 | 44.30 |
| SparK (Liao et al., 2026) | 23.83 | 35.48 | 46.86 | 42.31 | 35.67 | 21.08 | 17.76 | 21.17 | 23.26 | 57.75 | 89.36 | 35.49 | 9.80 | 69.62 | 45.89 | 40.17 | 38.47 |
| **GRIK** (Ours) | 25.14 | 43.13 | 55.60 | 48.60 | 48.48 | 24.49 | 29.47 | 23.63 | 26.48 | 64.50 | 91.51 | 43.05 | 10.58 | 72.50 | 64.05 | 56.11 | **45.46** |
| ***Channel pruning*** $r = 0.6$ ***(40% kept — aggressive pruning)*** | | | | | | | | | | | | | | | | | |
| $B$=128 | | | | | | | | | | | | | | | | | |
| Full KV | 22.18 | 23.90 | 47.77 | 46.11 | 45.99 | 23.34 | 21.07 | 22.56 | 20.00 | 40.00 | 90.82 | 40.60 | 10.87 | 72.00 | 59.43 | 51.21 | 39.87 |
| ThinK (Xu et al., 2025) | 19.83 | 25.23 | 41.08 | 44.48 | 47.08 | 22.13 | 21.61 | 22.17 | 19.54 | 39.50 | 89.55 | 39.76 | 11.50 | 72.50 | 57.58 | 50.53 | 39.00 |
| SparK (Liao et al., 2026) | 18.40 | 21.09 | 40.35 | 40.19 | 36.28 | 17.29 | 19.66 | 22.34 | 18.95 | 40.00 | 86.81 | 35.02 | 7.93 | 56.62 | 44.59 | 39.51 | 34.06 |
| **GRIK** (Ours) | 22.76 | 24.28 | 46.92 | 45.73 | 46.49 | 23.36 | 20.57 | 22.03 | 19.16 | 37.50 | 89.12 | 39.70 | 11.25 | 72.50 | 57.88 | 49.79 | **39.31** |
| $B$=512 | | | | | | | | | | | | | | | | | |
| Full KV | 24.91 | 34.59 | 53.27 | 46.71 | 47.55 | 24.48 | 24.72 | 22.97 | 23.90 | 53.00 | 91.93 | 43.09 | 10.67 | 72.50 | 64.04 | 55.76 | 43.38 |
| ThinK (Xu et al., 2025) | 21.11 | 29.64 | 48.14 | 46.76 | 48.36 | 23.67 | 24.06 | 22.00 | 21.95 | 42.00 | 89.96 | 40.34 | 11.10 | 72.50 | 60.21 | 52.45 | 40.89 |
| SparK (Liao et al., 2026) | 17.91 | 20.15 | 33.13 | 33.88 | 26.01 | 13.03 | 14.29 | 18.90 | 16.04 | 47.00 | 81.77 | 24.92 | 7.01 | 48.71 | 38.57 | 36.41 | 29.86 |
| **GRIK** (Ours) | 23.28 | 31.75 | 50.86 | 47.64 | 47.20 | 24.69 | 23.19 | 23.09 | 22.32 | 51.00 | 91.05 | 40.09 | 10.50 | 72.50 | 60.28 | 53.64 | **42.07** |
| $B$=1024 | | | | | | | | | | | | | | | | | |
| Full KV | 25.52 | 41.00 | 55.62 | 47.96 | 48.25 | 24.64 | 26.88 | 23.36 | 25.74 | 60.50 | 91.93 | 42.53 | 10.38 | 72.50 | 64.82 | 56.38 | 44.88 |
| ThinK (Xu et al., 2025) | 22.33 | 36.55 | 52.80 | 46.45 | 48.50 | 23.26 | 24.80 | 22.87 | 24.03 | 45.00 | 91.78 | 40.87 | 11.12 | 72.50 | 61.08 | 54.09 | 42.38 |
| SparK (Liao et al., 2026) | 13.38 | 14.25 | 25.52 | 25.74 | 19.02 | 12.07 | 12.63 | 16.67 | 14.21 | 45.25 | 69.31 | 17.62 | 9.11 | 35.81 | 36.31 | 34.99 | 25.12 |
| **GRIK** (Ours) | 25.16 | 38.40 | 53.13 | 48.08 | 47.84 | 24.23 | 24.77 | 22.92 | 24.04 | 55.50 | 91.26 | 40.99 | 11.33 | 72.50 | 62.18 | 54.38 | **43.54** |
| $B$=2048 | | | | | | | | | | | | | | | | | |
| Full KV | 25.71 | 43.82 | 55.95 | 48.46 | 48.32 | 24.44 | 29.63 | 23.67 | 26.63 | 66.00 | 91.37 | 43.20 | 9.93 | 72.50 | 65.08 | 57.00 | 45.73 |
| ThinK (Xu et al., 2025) | 24.49 | 41.72 | 54.80 | 48.20 | 48.55 | 24.59 | 26.98 | 23.19 | 25.31 | 48.50 | 91.87 | 41.73 | 10.67 | 72.50 | 61.56 | 53.72 | 43.65 |
| SparK (Liao et al., 2026) | 11.19 | 10.70 | 14.95 | 18.33 | 14.69 | 7.59 | 13.65 | 14.16 | 15.67 | 41.00 | 51.38 | 15.31 | 8.42 | 24.10 | 36.60 | 32.63 | 20.65 |
| **GRIK** (Ours) | 24.50 | 42.38 | 54.87 | 48.37 | 49.00 | 25.05 | 26.72 | 23.82 | 25.37 | 62.50 | 91.53 | 41.00 | 10.67 | 72.50 | 61.29 | 54.34 | **44.62** |

on the majority of tasks while paying $1.50\times$–$2.43\times$ less K-cache memory than ThinK at the same operating point.

# I Bootstrap Confidence Intervals on LongBench Avg

The main results in Table 2 report point estimates of the LongBench-v1 16-task average. All evaluations use greedy decoding (temperature 0) on fixed prompts, so cell-level run-to-run variance is zero. The relevant variability is across the 16 LongBench tasks themselves, whose per-task scores range from 0 to 95. We quantify this task-level variability via paired bootstrap resampling.

Table 16: Per-task LongBench on Qwen2.5-7B-Instruct, $w$=32, both pruning ratios. **Bold**: best per-block Avg; "–": not available. Full KV no-evict is $r$-independent (identical across sub-blocks).

| Method | Single-Doc QA | | | Multi-Doc QA | | | Summarization | | | Few-shot | | | Synth. | | Code | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NrtvQA | Qasper | MF-en | HotpotQA | 2WikiMQA | Musique | GovReport | QMSum | MultiNews | TREC | TriviaQA | SAMSum | PCount | PRe | Lcc | RB-P | Avg. |
| ***Channel pruning*** $r = 0.4$ ***(60% kept)*** | | | | | | | | | | | | | | | | | |
| $B$=128 | | | | | | | | | | | | | | | | | |
| Full KV | 23.58 | 29.61 | 38.28 | 46.91 | 40.66 | 23.03 | 19.50 | 21.44 | 17.78 | 38.50 | 77.57 | 43.07 | 8.00 | 98.00 | 52.09 | 55.30 | 39.58 |
| ThinK (Xu et al., 2025) | 23.90 | 29.30 | 36.48 | 47.18 | 41.11 | 21.39 | 19.62 | 21.59 | 17.66 | 38.50 | 79.46 | 43.31 | 8.00 | 97.00 | 51.42 | 54.60 | **39.41** |
| SparK (Liao et al., 2026) | 5.29 | 4.53 | 9.22 | 8.33 | 6.67 | 5.21 | 4.91 | 11.94 | 7.43 | 33.50 | 18.21 | 13.50 | 7.35 | 4.50 | 14.56 | 16.61 | 10.73 |
| **GRIK** (Ours) | 21.26 | 25.75 | 37.17 | 49.01 | 41.92 | 20.47 | 18.35 | 21.04 | 16.45 | 38.00 | 80.96 | 42.39 | 8.00 | 98.00 | 49.23 | 54.15 | 38.88 |
| $B$=512 | | | | | | | | | | | | | | | | | |
| Full KV | 24.54 | 34.99 | 45.27 | 49.57 | 42.12 | 23.46 | 22.40 | 22.07 | 21.01 | 39.50 | 84.61 | 44.61 | 8.00 | 98.00 | 56.58 | 60.92 | 42.35 |
| ThinK (Xu et al., 2025) | 24.99 | 34.09 | 44.84 | 50.40 | 42.46 | 24.68 | 22.04 | 21.94 | 20.70 | 40.00 | 84.17 | 44.15 | 8.00 | 97.50 | 55.00 | 59.60 | 42.16 |
| SparK (Liao et al., 2026) | 5.92 | 6.75 | 7.57 | 10.28 | 7.47 | 5.10 | 7.32 | 12.90 | 11.48 | 33.92 | 20.19 | 12.83 | 4.03 | 3.04 | 16.09 | 17.63 | 11.41 |
| **GRIK** (Ours) | 26.11 | 34.20 | 45.69 | 52.26 | 42.39 | 25.77 | 22.07 | 22.11 | 20.37 | 40.00 | 84.78 | 44.42 | 8.00 | 97.00 | 54.49 | 60.78 | **42.53** |
| $B$=1024 | | | | | | | | | | | | | | | | | |
| Full KV | 27.21 | 39.60 | 47.21 | 49.47 | 43.12 | 26.56 | 23.62 | 22.26 | 22.20 | 42.00 | 86.69 | 45.55 | 8.00 | 96.25 | 58.01 | 62.86 | 43.79 |
| ThinK (Xu et al., 2025) | 26.59 | 40.26 | 48.35 | 49.46 | 42.47 | 25.25 | 23.84 | 22.11 | 22.55 | 41.50 | 86.09 | 45.15 | 8.00 | 96.50 | 57.84 | 62.91 | 43.68 |
| SparK (Liao et al., 2026) | 5.77 | 6.56 | 8.99 | 10.18 | 6.42 | 4.17 | 8.47 | 12.81 | 13.01 | 34.00 | 19.03 | 13.23 | 5.12 | 2.45 | 16.51 | 17.16 | 11.49 |
| **GRIK** (Ours) | 27.25 | 38.62 | 48.27 | 52.43 | 43.26 | 26.52 | 23.95 | 22.03 | 21.69 | 44.00 | 86.27 | 45.10 | 8.00 | 97.00 | 56.70 | 62.59 | **43.98** |
| $B$=2048 | | | | | | | | | | | | | | | | | |
| Full KV | 27.20 | 41.28 | 48.97 | 50.96 | 44.15 | 27.85 | 25.19 | 22.66 | 22.92 | 49.50 | 88.29 | 45.94 | 8.00 | 92.75 | 59.01 | 63.50 | 44.89 |
| ThinK (Xu et al., 2025) | 27.44 | 42.00 | 50.50 | 51.99 | 44.14 | 27.82 | 26.32 | 22.55 | 22.83 | 47.00 | 88.25 | 44.95 | 8.00 | 96.00 | 58.13 | 64.17 | 45.13 |
| SparK (Liao et al., 2026) | 5.82 | 6.72 | 9.62 | 12.09 | 5.02 | 4.04 | 10.16 | 13.25 | 13.91 | 32.67 | 18.75 | 12.21 | 3.67 | 1.21 | 15.32 | 16.06 | 11.28 |
| **GRIK** (Ours) | 28.53 | 40.92 | 49.94 | 52.77 | 43.99 | 29.34 | 25.61 | 22.47 | 22.64 | 52.50 | 86.15 | 44.85 | 8.00 | 95.00 | 58.32 | 63.60 | **45.29** |
| ***Channel pruning*** $r = 0.6$ ***(40% kept — aggressive pruning)*** | | | | | | | | | | | | | | | | | |
| $B$=128 | | | | | | | | | | | | | | | | | |
| Full KV | 23.58 | 29.61 | 38.28 | 46.91 | 40.66 | 23.03 | 19.50 | 21.44 | 17.78 | 38.50 | 77.57 | 43.07 | 8.00 | 98.00 | 52.09 | 55.30 | 39.58 |
| ThinK (Xu et al., 2025) | 23.18 | 29.31 | 36.28 | 48.53 | 41.17 | 21.71 | 19.75 | 21.27 | 17.25 | 38.50 | 80.82 | 40.79 | 8.00 | 96.00 | 52.07 | 55.81 | **39.40** |
| SparK (Liao et al., 2026) | 4.32 | 5.78 | 8.65 | 7.95 | 6.39 | 5.06 | 4.70 | 11.64 | 7.14 | 32.75 | 19.12 | 12.14 | 5.08 | 4.14 | 16.18 | 17.74 | 10.55 |
| **GRIK** (Ours) | 21.88 | 26.70 | 35.39 | 48.02 | 42.08 | 21.13 | 17.82 | 20.88 | 16.26 | 38.50 | 80.87 | 40.78 | 8.00 | 91.50 | 48.87 | 53.64 | 38.27 |
| $B$=512 | | | | | | | | | | | | | | | | | |
| Full KV | 24.54 | 34.99 | 45.27 | 49.57 | 42.12 | 23.46 | 22.40 | 22.07 | 21.01 | 39.50 | 84.61 | 44.61 | 8.00 | 98.00 | 56.58 | 60.92 | 42.35 |
| ThinK (Xu et al., 2025) | 23.66 | 33.62 | 43.29 | 50.64 | 44.73 | 23.48 | 22.22 | 21.88 | 20.60 | 39.50 | 82.59 | 42.89 | 8.00 | 97.50 | 56.15 | 60.17 | 41.93 |
| SparK (Liao et al., 2026) | 4.79 | 6.58 | 7.63 | 8.35 | 4.78 | 3.98 | 7.82 | 12.23 | 13.11 | 32.25 | 18.05 | 11.02 | 5.97 | 1.12 | 17.96 | 19.24 | 10.93 |
| **GRIK** (Ours) | 25.41 | 33.44 | 43.92 | 52.84 | 43.78 | 25.64 | 21.37 | 22.17 | 19.87 | 40.00 | 85.49 | 42.43 | 8.00 | 94.50 | 54.26 | 59.72 | **42.05** |
| $B$=1024 | | | | | | | | | | | | | | | | | |
| Full KV | 27.21 | 39.60 | 47.21 | 49.47 | 43.12 | 26.56 | 23.62 | 22.26 | 22.20 | 42.00 | 86.69 | 45.55 | 8.00 | 96.25 | 58.01 | 62.86 | 43.79 |
| ThinK (Xu et al., 2025) | 26.03 | 39.04 | 48.61 | 50.22 | 44.21 | 26.26 | 24.18 | 22.37 | 21.54 | 42.50 | 85.65 | 43.06 | 8.00 | 97.00 | 57.45 | 62.08 | **43.64** |
| SparK (Liao et al., 2026) | 4.52 | 4.99 | 6.88 | 8.55 | 6.72 | 3.94 | 9.92 | 11.99 | 14.57 | 32.17 | 17.73 | 11.54 | 4.78 | 0.59 | 17.97 | 18.81 | 10.98 |
| **GRIK** (Ours) | 25.63 | 37.47 | 49.48 | 53.90 | 44.03 | 26.77 | 23.16 | 22.00 | 21.35 | 44.50 | 85.75 | 42.22 | 8.00 | 95.00 | 55.47 | 61.09 | 43.49 |
| $B$=2048 | | | | | | | | | | | | | | | | | |
| Full KV | 27.20 | 41.28 | 48.97 | 50.96 | 44.15 | 27.85 | 25.19 | 22.66 | 22.92 | 49.50 | 88.29 | 45.94 | 8.00 | 92.75 | 59.01 | 63.50 | 44.89 |
| ThinK (Xu et al., 2025) | 26.40 | 41.24 | 49.94 | 51.66 | 45.27 | 28.32 | 26.59 | 22.56 | 22.53 | 47.50 | 87.12 | 42.64 | 8.00 | 97.00 | 58.65 | 63.27 | **44.92** |
| SparK (Liao et al., 2026) | 3.85 | 5.69 | 8.01 | 9.76 | 4.70 | 3.60 | 12.20 | 12.30 | 14.92 | 32.00 | 15.14 | 10.35 | 4.56 | 1.12 | 17.85 | 17.65 | 10.86 |
| **GRIK** (Ours) | 28.17 | 41.08 | 49.72 | 52.34 | 44.73 | 27.74 | 25.05 | 22.52 | 21.94 | 51.50 | 85.63 | 42.33 | 8.00 | 95.50 | 56.74 | 62.73 | 44.73 |

**Procedure.** For each (model, KV budget, ratio) cell, we resample the 16 task indices with replacement and recompute the AVG. We use $N$=10,000 iterations and report 95% percentile-method confidence intervals ($\alpha$=0.05). When comparing GRIK to a baseline at a matched cell, the *same* resampled task indices are used on both sides, so the gap CI accounts for cross-task correlation. This produces three quantities per cell pair: the gap mean (GRIK − baseline), the 95% CI on the gap, and the bootstrap probability $P(\text{GRIK} > \text{baseline})$. Per-cell CIs span roughly $\pm$10 points around the mean, reflecting the wide per-task score distribution; the paired gap CIs are an order of magnitude tighter (often $\pm$1–3 points) because cross-task correlation is removed. The gap is the metric of inferential interest.

**Summary.** Across all (model, budget, $r$) cells:

Table 17: **Per-cell LongBench AVG with 95% bootstrap CI.** $N$=10,000 task-level resamples; CI half-width in brackets.

| Backbone ($r$) | Method | $B$=128 | $B$=512 | $B$=1024 | $B$=2048 | no-evict |
|---|---|---|---|---|---|---|
| LLaMA-3-8B (0.4) | ThinK | $35.6 \pm 10.5$ | — | — | — | — |
| | ThinK-mean | $35.4 \pm 10.3$ | $37.0 \pm 10.5$ | $38.1 \pm 10.7$ | $39.5 \pm 10.5$ | $41.8 \pm 10.7$ |
| | SparK | $32.8 \pm 10.5$ | $33.9 \pm 10.5$ | $34.7 \pm 10.4$ | $34.9 \pm 10.8$ | — |
| | GRIK | $36.0 \pm 10.6$ | $38.6 \pm 10.7$ | $40.1 \pm 10.8$ | $41.3 \pm 11.1$ | $42.2 \pm 10.8$ |
| LLaMA-3-8B (0.6) | ThinK | $35.1 \pm 10.3$ | — | — | — | — |
| | SparK | $28.0 \pm 10.1$ | $26.7 \pm 9.9$ | $24.1 \pm 9.7$ | $21.9 \pm 8.9$ | — |
| | GRIK | $35.0 \pm 10.4$ | $37.5 \pm 10.4$ | $38.3 \pm 10.6$ | $39.2 \pm 10.7$ | — |
| LLaMA-3.1-8B (0.4) | Full KV | $40.0 \pm 10.3$ | $43.5 \pm 10.2$ | $44.9 \pm 10.2$ | $45.7 \pm 10.3$ | — |
| | SparK | $38.1 \pm 9.9$ | $40.0 \pm 9.9$ | $39.9 \pm 9.7$ | $38.4 \pm 10.0$ | $33.5 \pm 9.6$ |
| | GRIK | $39.6 \pm 10.3$ | $43.2 \pm 10.2$ | $44.6 \pm 10.3$ | $45.5 \pm 10.3$ | $46.2 \pm 10.1$ |
| LLaMA-3.1-8B (0.6) | SparK | $34.0 \pm 8.9$ | $29.9 \pm 8.7$ | $25.1 \pm 7.4$ | $20.7 \pm 6.1$ | $16.8 \pm 5.2$ |
| | GRIK | $39.3 \pm 9.8$ | $42.1 \pm 10.2$ | $43.6 \pm 10.1$ | $44.7 \pm 10.2$ | $45.7 \pm 10.2$ |
| Qwen-2.5-7B (0.4) | Full KV | $39.5 \pm 10.9$ | $42.3 \pm 11.4$ | $43.8 \pm 11.3$ | $44.9 \pm 10.9$ | — |
| | ThinK | $39.4 \pm 11.1$ | $42.0 \pm 11.2$ | $43.7 \pm 11.1$ | $45.2 \pm 11.2$ | — |
| | GRIK | $38.8 \pm 11.5$ | $42.5 \pm 11.2$ | $44.0 \pm 11.2$ | $45.3 \pm 10.9$ | — |
| Qwen-2.5-7B (0.6) | SparK | $10.6 \pm 3.6$ | $10.9 \pm 3.7$ | $11.0 \pm 3.8$ | $10.9 \pm 3.7$ | $10.1 \pm 3.9$ |
| | GRIK | $38.3 \pm 10.8$ | $42.1 \pm 11.2$ | $43.4 \pm 11.3$ | $44.7 \pm 11.3$ | $46.6 \pm 11.2$ |

Table 18: **Paired bootstrap gaps vs. SparK** (GRIK − SparK). Format: $\Delta\pm$ half-width of 95% CI; all cells exclude zero ($P$=1.00).

| Backbone ($r$) | $B$=128 | $B$=512 | $B$=1024 | $B$=2048 | no-evict |
|---|---|---|---|---|---|
| LLaMA-3-8B (0.4) | $+3.2 \pm 1.6$ | $+4.7 \pm 2.7$ | $+5.5 \pm 2.7$ | $+6.5 \pm 2.9$ | — |
| LLaMA-3-8B (0.6) | $+7.1 \pm 3.3$ | $+10.9 \pm 4.2$ | $+14.4 \pm 4.4$ | $+17.3 \pm 4.1$ | — |
| LLaMA-3.1-8B (0.4) | $+1.4 \pm 0.9$ | $+3.2 \pm 1.5$ | $+4.7 \pm 2.0$ | $+7.0 \pm 2.5$ | $+12.7 \pm 2.8$ |
| LLaMA-3.1-8B (0.6) | $+5.3 \pm 2.4$ | $+12.2 \pm 3.2$ | $+18.4 \pm 4.6$ | $+24.0 \pm 6.1$ | $+28.9 \pm 8.0$ |
| Qwen-2.5-7B (0.6) | $+27.8 \pm 10.2$ | $+31.1 \pm 11.4$ | $+32.6 \pm 11.3$ | $+33.9 \pm 11.3$ | $+36.5 \pm 11.1$ |

- **vs. SparK**: GRIK strictly dominates in every cell across all three backbones, all budgets, and both ratios. Gap means range from $+1.4$ to $+36.5$, all with 95% CIs excluding zero ($P$=1.00).
- **vs. ThinK-mean** (LLaMA-3-8B; the GQA-aggregated reformulation that shares GRIK's storage layout): GRIK strictly dominates at $B \in \{512, 1024, 2048\}$, $r$=0.4 with 95% CI excluding zero ($P$=1.00); the closest matched cell ($B$=128) yields $+0.47$ with $P$=0.92.
- **vs. ThinK** (native per-query-head): on LLaMA-3-8B at $r$=0.4, $B$=128 the gap is $+0.33$ with $P$=0.81 (statistically tied at this budget); the structural advantage of GRIK is the $4\times$ K-cache footprint reduction ($0.60\times$ vs. $2.40\times$ Full KV), not LongBench AVG at this single matched cell.
- **vs. Full KV**: GRIK is statistically indistinguishable from Full KV on LLaMA-3.1-8B at $r$=0.4 ($\Delta \in [-0.27, -0.19]$ across budgets, gap CIs spanning at most 0.5 points). On Qwen-2.5 at $r$=0.4, GRIK matches or slightly exceeds Full KV at $B \in \{512, 1024, 2048\}$.

Tables 17, 18, and 19 report per-cell means with 95% CIs and paired gap CIs. ✓ marks gap 95% CIs that exclude zero. The vs. SparK comparison is consolidated separately because every cell is significant; gaps against Full KV, ThinK, and ThinK-mean are reported in Table 19.

## J THINK-MEAN VARIANT ABLATION

This section reports a score-form ablation that controls for storage layout. The native ThinK formulation (Xu et al., 2025) assigns one mask per query head, yielding $H_q$=32 effective K-cache heads

Table 19: **Paired bootstrap gaps vs. Full KV, ThinK, ThinK-mean.** Format: $\Delta\pm$ half-width of 95% CI. ✓ marks 95% CI excluding zero.

| Backbone ($r$) | vs. | $B$=128 | $B$=512 | $B$=1024 | $B$=2048 | no-evict |
|---|---|---|---|---|---|---|
| LLaMA-3-8B (0.4) | ThinK | $+0.3 \pm 0.7$ | — | — | — | — |
| | ThinK-mean | $+0.5 \pm 0.7$ | $+1.7 \pm 1.4$ ✓ | $+2.0 \pm 1.8$ ✓ | $+1.8 \pm 2.1$ ✓ | $+0.3 \pm 0.3$ |
| LLaMA-3-8B (0.6) | ThinK | $-0.1 \pm 0.4$ | — | — | — | — |
| LLaMA-3.1-8B (0.4) | Full KV | $-0.3 \pm 0.2$ ✓ | $-0.2 \pm 0.3$ | $-0.3 \pm 0.2$ ✓ | $-0.3 \pm 0.3$ ✓ | — |
| Qwen-2.5-7B (0.4) | Full KV | $-0.7 \pm 0.9$ | $+0.2 \pm 0.6$ | $+0.2 \pm 0.5$ | $+0.4 \pm 0.6$ | — |
| | ThinK | $-0.5 \pm 0.7$ | $+0.4 \pm 0.3$ ✓ | $+0.3 \pm 0.6$ | $+0.2 \pm 0.8$ | — |

Table 20: **ThinK-mean variant on LLaMA-3-8B-Instruct.** LongBench 16-task AVG. ThinK is the native per-query-head variant (cited from Xu et al. (2025) Table 2); ThinK-mean is the GQA-aggregated reformulation sharing GRIK's storage layout. "–": not measured.

| **Method** | **Storage** | $r$ | $B$=128 | $B$=512 | $B$=1024 | $B$=2048 |
|---|---|---|---|---|---|---|
| ThinK[a] | $H_q$=32 | 0.4 | 35.63 | 37.39 | 39.00 | 40.05 |
| ThinK-mean (8h) | $H_{kv}$=8 | 0.4 | 35.45 | 36.96 | 38.11 | 39.48 |
| **GRIK** (8h, ours) | $H_{kv}$=8 | 0.4 | **35.92** | **38.63** | **40.12** | **41.30** |
| ThinK[a] | $H_q$=32 | 0.6 | 35.06 | 36.44 | **38.57** | 38.51 |
| ThinK-mean (8h) | $H_{kv}$=8 | 0.6 | 34.91 | 36.30 | 37.40 | 38.30 |
| **GRIK** (8h, ours) | $H_{kv}$=8 | 0.6 | **35.00** | **37.45** | 38.45 | **39.24** |

[a] Cited from Xu et al. (2025) Table 2.

on LLaMA-3-8B regardless of GQA, a violation of condition C1 (§3). To isolate the score form from this layout difference, we construct **ThinK-mean (8h)**: the per-query-head ThinK score is averaged uniformly within each KV group, yielding a one-shot mask defined per KV head, with the same $H_{kv}$=8 K-cache footprint as GRIK. ThinK-mean and GRIK therefore use identical storage; the only remaining difference is GRIK's iterative $\alpha_h$-weighted aggregation with the static $\|W_k\|$ prior. Comparing the three side-by-side decomposes the (storage, score-form) decision into its two axes.

Three readings emerge from Table 20.

**Storage axis (ThinK → ThinK-mean 8h).** Reformulating ThinK to honor C1 by averaging within each KV group costs little accuracy (typically $-0.2$ to $-0.6$ pp at $r$=0.4) while reducing K-cache storage by $4\times$. The score form is essentially the same; only the storage layout changes. This isolates the cost of enforcing the GQA constraint.

**Score-form axis (ThinK-mean 8h → GRIK 8h).** With storage held fixed at $H_{kv}$=8, replacing one-shot mean aggregation with GRIK's $\alpha_h$-weighted iterative refinement and static $\|W_k\|$ prior recovers and exceeds the original ThinK accuracy on every $r$=0.4 cell. The gap widens with budget, from $+0.47$ at $B$=128 to $+1.82$ at $B$=2048.

**Combined effect (ThinK → GRIK).** GRIK matches or exceeds ThinK on the majority of cells while using $1/4$ of the K-cache storage. At $r$=0.4, $B$=2048, GRIK is $+1.25$ pp ahead of ThinK at $4\times$ less storage. This is the practical bottom line of the ablation: the storage and the score form are separable design choices, and GRIK is favorable on both.

## K Per-Task RULER Results

Table 21 summarizes the aggressive setting. Table 22 reports its per-task RULER scores for GRIK and SparK on LLaMA-3.1-8B-Instruct at KV budget 200K (channel-only regime). The eight needle-in-a-haystack subtasks (`niah_*`) score 0.0% under the evaluated SparK recovery implementation at every length; this result does not rule out other ways to store or reconstruct token-dependent channels.

Table 21: Aggressive RULER-13 stress test on LLaMA-3.1-8B-Instruct at $r$=0.7 (30% of K channels kept), KV budget 200K; no token eviction. This setting is distinct from the primary $r$=0.4 result in Table 8.

| Method | 4K | 8K | 16K | 32K | Avg. |
|---|---|---|---|---|---|
| Full KV | 95.10 | 93.10 | 90.20 | 86.00 | 91.10 |
| ThinK (Xu et al., 2025) | 58.30 | 39.20 | 37.50 | 36.40 | 42.85 |
| SparK (Liao et al., 2026) | 6.17 | 3.80 | 3.32 | 2.58 | 3.97 |
| GRIK | 61.52 | 60.00 | 56.70 | 54.24 | 58.12 |

Table 22: Per-task RULER scores (%) at KV 200K, $r$=0.7, LLaMA-3.1-8B-Instruct.

| | **GRIK (Ours)** | | | | **SparK (Liao et al., 2026)** | | | |
|---|---|---|---|---|---|---|---|---|
| **Task** | 4K | 8K | 16K | 32K | 4K | 8K | 16K | 32K |
| niah_single_1 | 94.00 | 90.00 | 82.00 | 63.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| niah_single_2 | 70.50 | 66.50 | 75.50 | 64.50 | 0.00 | 0.00 | 0.00 | 0.00 |
| niah_single_3 | 54.00 | 61.50 | 57.00 | 65.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| niah_multikey_1 | 82.00 | 89.00 | 85.00 | 73.50 | 0.00 | 0.00 | 0.00 | 0.00 |
| niah_multikey_2 | 77.50 | 75.00 | 68.50 | 67.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| niah_multikey_3 | 36.50 | 27.00 | 17.00 | 13.50 | 0.00 | 0.00 | 0.00 | 0.00 |
| niah_multivalue | 75.50 | 76.50 | 83.62 | 82.12 | 0.00 | 0.00 | 0.00 | 0.00 |
| niah_multiquery | 87.75 | 90.88 | 82.38 | 90.50 | 0.00 | 0.00 | 0.00 | 0.00 |
| vt | 9.90 | 7.70 | 5.40 | 10.70 | 0.00 | 0.00 | 0.00 | 0.00 |
| cwe | 27.75 | 3.40 | 0.05 | 0.00 | 21.35 | 9.95 | 5.50 | 1.60 |
| fwe | 51.83 | 65.50 | 63.67 | 62.83 | 22.83 | 15.00 | 14.17 | 14.50 |
| qa_1 | 76.50 | 72.00 | 69.50 | 66.50 | 21.00 | 11.50 | 12.50 | 9.00 |
| qa_2 | 56.00 | 55.00 | 47.50 | 46.00 | 15.00 | 13.00 | 11.00 | 8.50 |
| **Avg.** | **61.52** | **60.00** | **56.70** | **54.24** | 6.17 | 3.80 | 3.32 | 2.58 |

GRIK retains nonzero retrieval performance in this aggressive regime but declines on several subtasks as context grows.

## L TRT-LLM KV Pool Validation

The total KV-pool memory numbers reported in §5.3 and Table 1 are computed from the standard baseline layout formula

$$\text{B/tok/layer} \;=\; H_{\text{kv}} \cdot d \cdot b_K \;+\; H_{\text{kv}} \cdot d \cdot b_V, \tag{11}$$

where $b_K, b_V$ are the per-element byte widths of the K and V tensors (2 for bf16, 1 for FP8). For GRIK, $d$ is replaced by $d_{\text{keep}}$ only in the K term. For the native ThinK-on-GQA layout plotted in Figure 1, $H_{\text{kv}}$ is replaced by $H_q$ in both the K and V terms, while $d$ is replaced by $d_{\text{keep}}$ only in the K term. Tables 23 and 24 validate the unpruned pool allocation and FP8 byte width; these allocator-only checks are separate from the fused GRIK H100 serving benchmark in Table 5.

To confirm that this formula reflects what a real serving stack actually allocates — not a derivation that overlooks padding, block alignment, or pool reservation overhead — we cross-check against TRT-LLM 1.0.0's PyTorch backend allocator log on LLaMA-3.1-8B-Instruct.

**Method.** We launch the baseline (Full KV) configuration under two different `max_seq_len` settings and capture the `[MemUsageChange]` events that TRT-LLM emits on KV-pool construction. For example,

```
[TensorRT-LLM][INFO] [MemUsageChange] Allocated 1.16 GiB
for max tokens in paged KV cache (9472).
```

Table 23: TRT-LLM 1.0.0 PyTorch backend KV-pool size: allocator log vs. layout formula, LLaMA-3.1-8B-Instruct, bf16. Formula matches within $0.4\%$ (below paged-KV rounding granularity).

| max_tokens | TRT-LLM logged | Formula (Eq. 11) | $\Delta$ |
|---|---|---|---|
| 8 224 | 1.000 GiB | 1.004 GiB | $+0.4\%$ |
| 9 472 | 1.160 GiB | 1.156 GiB | $-0.3\%$ |

Table 24: FP8 KV pool on LLaMA-3.1-8B-Instruct. Same Hopper-class GPU and TRT-LLM build as Table 23.

| KV dtype | Logged size | max_tokens | B/tok/layer |
|---|---|---|---|
| bf16 | 1.000 GiB | 8 224 | 4096 |
| FP8 | 0.500 GiB | 8 256 | 2048 |

yields the (logged size, max-tokens) pair $(1.16\ \text{GiB},\ 9472)$. We compare each logged size to the value predicted by Equation 11 multiplied by the number of layers (32) and the reported `max_tokens`.

**Result (bf16).** The deviation in both cases is below $0.5\%$ and arises from sub-block rounding inside TRT-LLM's paged KV-cache manager (blocks are sized to align with the block size and number of KV heads). The rounded allocator log is consistent with the baseline data-byte formula at these two settings; it does not isolate every allocator overhead.

**Result (FP8 KV).** TRT-LLM 1.0 supports FP8 KV-cache storage via `KvCacheConfig(dtype="fp8")`. We re-run the same LLaMA-3.1 baseline on the same Hopper-class GPU and capture the allocator log:

The logged baseline FP8 pool is approximately half the bf16 pool, consistent with a one-byte rather than a two-byte element. An asymmetric GRIK+FP8 pool was not measured in this validation.

**Scope of allocator validation.** Tables 23 and 24 measure only unmodified Full-KV allocations. Substituting a narrower K width into Equation 11 predicts the GRIK data footprint, but these baseline logs do not establish formula-to-allocator agreement for a modified asymmetric-K/V runtime. The separate fused H100 GRIK serving measurement is reported in Table 5 at $r{=}0.5$.