STACKV: SPATIO-TEMPORALLY AWARE COMPRESSION OF KV CACHES IN HYBRID LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Hybrid language models, which interleave linear recurrent layers with full-attention layers, offer a promising balance between training efficiency and expressive power. However, during autoregressive generation, the key-value (KV) cache of the attention layers also poses a severe memory bottleneck. Existing KV cache eviction strategies, primarily designed for pure Transformers, are sub-optimal for hybrid architectures. They rely exclusively on signals from full-attention layers, without accounting for recurrent-state dynamics, and employ unconstrained global pruning, which disrupts the contiguous reasoning chains required for complex tasks. In this paper, we introduce **StacKV**, a training-free KV cache compression framework tailored for hybrid models, addressing both *what* to retain and *where* to retain it. First, we identify a complementary retention signal from recurrent-state dynamics: tokens with low-magnitude recurrent updates are often overlooked by attention-based KV selection, yet retaining them alongside high-attention tokens improves performance. Second, to prevent the compounding loss of local context during rolling eviction, we propose an age-aware non-uniform spatial blocking strategy. Extensive experiments on 5 tasks demonstrate that StacKV outperforms existing baselines, maintaining over 95% of full-cache performance on average for each model at a 25% retention budget, while achieving a 1.69× serving throughput speedup in Nano-vLLM.

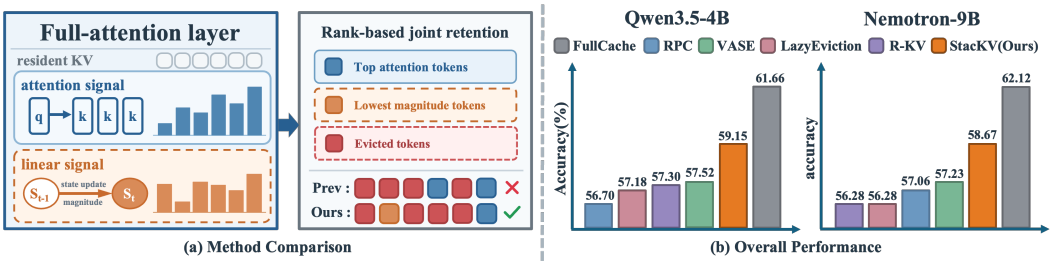

Figure 1: **(a)** Complementary temporal views for KV retention: recent-query attention (blue) assesses current access, while linear-update magnitudes (orange) record processing-time state transitions. StacKV combines both within cache blocks. **(b)** Mean scores over five benchmarks and five retention ratios; FullCache is averaged over benchmarks. All averages use equal weights.

1 INTRODUCTION

Large language models support diverse language understanding and generation tasks (Team, 2024; Yang et al., 2025a). During autoregressive decoding, full-attention layers cache previous tokens' keys and values to avoid redundant computation (Pope et al., 2022); this key-value (KV) cache grows linearly with sequence length and batch size, creating a memory bottleneck (Kwon et al., 2023). Prior work reduces this cost by retaining a compact subset of historical KV entries (Zhang et al., 2023; Cai et al., 2025). Hybrid architectures reduce memory by interleaving full-attention and linear-attention layers, maintaining token-wise KV caches in only part of the network (Yang et al., 2025b; Team et al., 2025). However, the remaining full-attention caches still grow with sequence length and

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.

batch size, leaving room for further savings during long generation. In our Qwen experiments, 25% retention raises the reported Nano-vLLM batch size from 77 to 192, achieving $1.69\times$ FullCache throughput at these batch sizes (Table 3).

Existing KV-cache compression methods primarily rely on signals from full-attention layers to decide which tokens to retain. This induces two critical misalignments when applied to hybrid models. First, the pruning is guided solely by scores from full-attention layers, completely detached from the concurrent recurrent states (S_t) in adjacent linear layers. Second, under unconstrained global Top-K ranking, older tokens undergo compounding rounds of eviction, eventually carving contiguous temporal voids that sever context dependency and degrade multi-step reasoning performance.

To resolve these challenges, we introduce **StacKV** (**S**patio-**t**emporally-**a**ware **c**ompression), a training-free eviction framework that coordinates explicit cache retention with implicit recurrent dynamics. StacKV rests on two core mechanisms:

First, to foster cross-memory synergy, StacKV leverages linear update dynamics ($\Delta S_t = S_t - S_{t-1}$) to guide cache retention. Because low-magnitude recurrent updates identify useful candidates beyond attention-based selection, StacKV repurposes the full-attention cache as a compensatory buffer by favoring both high-attention tokens and tokens with low recurrent update magnitudes.

Second, to bound temporal voids, StacKV counteracts the compounding sparsity of rolling eviction via an *age-aware block schedule*. Because older context experiences repeated pruning across decoding steps, uniform block constraints allow distant timeline gaps to expand unchecked. StacKV enforces fine-grained candidate blocks over distant history and coarser blocks over recent context, establishing an explicit structural bound on contiguous deletions within the current candidate sequence at each compression round.

StacKV extracts recurrent update norms on-the-fly during inference with near-zero scalar caching overhead. To eliminate serving latency penalties, we implement a fused eviction kernel that natively executes block scoring and non-contiguous gathers. Evaluated across challenging reasoning benchmarks (including AIME, HMMT, GPQA, and LiveCodeBench), StacKV retains competitive parity with full-cache baselines under a **25%** retention budget on hybrid models such as Qwen3.5-4B, outperforming existing attention-only pruning approaches. Our main contributions are summarized as follows:

- To the best of our knowledge, we are the first to use low-magnitude recurrent-update scores from linear layers for training-free KV-cache compression in pretrained hybrid models, extending token selection beyond full-attention signals.
- We propose StacKV, a training-free method that combines attention and recurrent signals within age-aware block quotas, using finer blocks for older candidates to preserve positional coverage while selecting tokens under a fixed budget.
- We develop fused scoring and batched selection kernels that keep the added inference overhead low. Across three hybrid architectures, StacKV improves accuracy over the evaluated baselines at matched retention budgets.

2 Related Work

2.1 KV-Cache Compression

Training-free KV eviction includes H_2O's recency and accumulated attention "heavy hitters" (Zhang et al., 2023), and StreamingLLM's initial attention sinks and recent window (Xiao et al., 2024). SnapKV selects head-specific salient positions through observation-window attention (Li et al., 2024), while PyramidKV varies layer budgets (Cai et al., 2025). Quest selects query-relevant pages (Tang et al., 2024b). Head-aware policies include Ada-KV's adaptive budgets (Feng et al., 2025), and RazorAttention's retrieval-head distinction (Tang et al., 2024a). Quantization reduces entry size through KIVI's asymmetric per-channel/per-token scheme (Liu et al., 2024), KVQuant's outlier-aware, non-uniform quantization (Hooper et al., 2025), and DesireKV's joint eviction–quantization policy based on token importance and quantization sensitivity (Cheng et al., 2026). During decoding, RPC periodically compresses caches using recent-query attention (Song et al., 2025), while R-KV balances attention importance against key redundancy (Cai et al., 2026). VaSE protects

large-magnitude values and introduces stochastic eviction (Chang et al., 2026); LazyEviction delays removal to capture recurring attention importance (Zhang et al., 2025). StacKV supplements recent-query attention with recorded recurrent updates and allocates retention across cache positions.

2.2 Linear and Hybrid Attention Architectures

Transformer caches grow with sequence length (Vaswani et al., 2017), whereas linear attention maintains fixed-size states (Katharopoulos et al., 2020). Recurrent designs use retention (Sun et al., 2023), selective state spaces (Gu & Dao, 2024; Dao & Gu, 2024), and gating (Yang et al., 2024). DeltaNet uses delta-rule updates (Yang et al., 2025c), and Gated DeltaNet adds retention gates (Yang et al., 2025b). Kimi Linear uses finer-grained gated delta updates (Team et al., 2025). Finite recurrent states can face exact-retrieval challenges (Arora et al., 2023). Hybrids retain attention for retrieval: Griffin pairs gated recurrences with local attention (De et al., 2024), Jamba interleaves Transformer and Mamba (Lieber et al., 2024), Samba uses Mamba and sliding-window attention (Ren et al., 2025), and hybrid DeltaNet adds local or global attention (Yang et al., 2025c). Hymba mixes attention/state-space heads with cross-layer KV sharing (Dong et al., 2024), while Kimi Linear interleaves linear and full attention (Team et al., 2025).

3 Temporal Complementarity and Spatial Coverage

We examine temporal complementarity by first comparing selection signals with a fixed linear source, then varying the source under the selected rule. We separately test spatial coverage by varying block allocation with attention scores alone. For hybrid selection, the retention-budget share assigned to linear-update scores is denoted by $\alpha \in [0, 1]$; attention receives the remaining share $1 - \alpha$. All compression experiments and token-selection diagnostics in this section use a retention ratio of $\rho = 1/4$. We evaluate retention quality on 500 sampled PolyMath (Wang et al., 2025) problems using Qwen3.5-4B (Yang et al., 2025a) and NVIDIA-Nemotron-Nano-9B-v2 (NVIDIA et al., 2025). We report ave@4: correctness averaged over four sampled responses per problem and then across problems.

3.1 Temporal Complementarity from Linear Updates

To supplement recent-query attention, we record each token's recurrent-update magnitude when it is processed for use in later compression. Measuring the overall state change provides a common metric across linear layers without relying on architecture-specific update components. Let $S_{t,\ell,h}$ be the state of head h in linear layer ℓ after processing token t, with H_ℓ heads per layer. Its update magnitude is

$$u_{t,\ell} = \frac{1}{H_\ell} \sum_{h=1}^{H_\ell} \|S_{t,\ell,h} - S_{t-1,\ell,h}\|_F \,, \tag{1}$$

where $\|\cdot\|_F$ is the Frobenius norm over the state dimensions. The architecture-specific updates are given in Appendix A.

For the selection-signal comparisons, each full-attention layer takes update magnitudes from its nearest preceding linear layer. This source assignment stays fixed while the selection rule varies.

We compare attention, high-magnitude, and low-magnitude token selections on subsets of GPQA (Rein et al., 2024) and MATH-500 (Lightman et al., 2023), using Qwen3.5-4B and NVIDIA-Nemotron-Nano-9B-v2. Low-magnitude selection overlaps less with attention at every displayed layer (Figure 2, left): the layer-averaged overlaps are 6.5% and 12.1% for the two models, versus 27.4% and 38.6% for high-magnitude selection.

Viewed over the token history (Figure 2, middle), low-magnitude selection extends beyond the regions prioritized by recent-query attention, whereas high-magnitude selection overlaps more with attention-selected tokens. This suggests that recorded linear updates can help preserve key historical information overlooked by recent-query attention.

Under the PolyMath setup above, mixtures use $\alpha = 0.5$, selecting by attention first, then by linear scores among unselected candidates. Low-magnitude selection improves ave@4 over attention

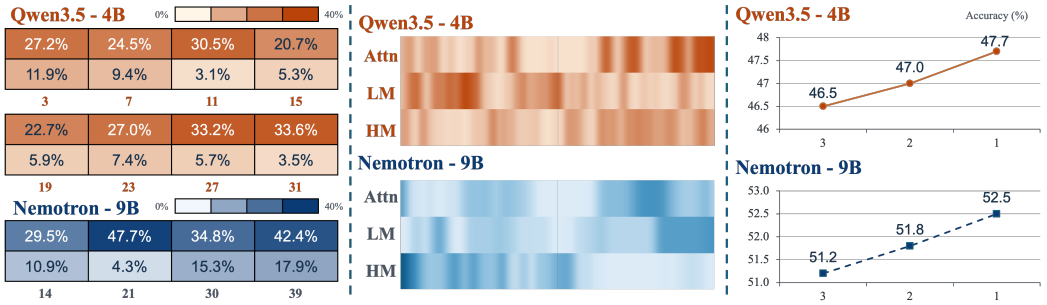

Figure 2: **Left:** Overlap with attention-selected tokens at the labeled full-attention layers; each row pair shows high-magnitude selection above low-magnitude selection. **Middle:** Positional distributions of attention (Attn.), low-magnitude (LM), and high-magnitude (HM) selections, from earlier to later tokens. **Right:** PolyMath ave@4 (%) using the third-nearest (3), second-nearest (2), or nearest (1) preceding linear layer as the score source, with a fixed $\alpha = 0.5$.

alone by 1.5/1.2 percentage points for Qwen/Nemotron, whereas high-magnitude selection hurts performance (Table 1).

We replace linear-score selection with uniform random, low-attention, or position-matched random selection (Attn./Pos-Matched) at fixed α. Pos-Matched samples uniformly without replacement, matching Low-Mag's token count in each 10% position bin at every compression. Low-attention and Pos-Matched underperform attention alone in both models. Random selection improves Qwen by 0.1 points but trails Low-Mag by 1.4 points. Low-Mag exceeds Pos-Matched by 2.0/1.5 percentage points for Qwen/Nemotron, indicating that token identity matters beyond coarse position.

Table 1: PolyMath ave@4 (%) on 500 sampled problems at $\rho = 1/4$. All mixtures use $\alpha = 0.5$; spatial-coverage policies use attention scores only.

Policy	Qwen3.5-4B	Nemotron-9B
Baseline		
Attn. ($\alpha = 0$)	46.2%	51.3%
Magnitude tails ($\alpha = 0.5$)		
Attn./High-Mag	44.5%	50.3%
Attn./Low-Mag	**47.7%**	**52.5%**
Selection controls ($\alpha = 0.5$)		
Attn./Random	46.3%	50.1%
Attn./Low-Attn	45.4%	50.8%
Attn./Pos-Matched	45.7%	51.0%
Spatial coverage (attention only)		
all-128 blocks	46.6%	51.7%
256/128/64 blocks	45.6%	51.1%
64/128/256 blocks	**47.1%**	**52.1%**

To understand what Low-Mag contributes beyond attention, we examine which token types it retains. At $\rho = 1/4$, the share of formula-like tokens is 17.12–22.61 percentage points higher with Low-Mag than with attention/RPC on AIME 2024 and GPQA across both models (Appendix C). The geometry example (Appendix I) illustrates their value for later reasoning: Low-Mag preserves earlier formula details that StacKV reuses correctly, whereas RPC substitutes an incorrect relation. These observations suggest that Low-Mag complements attention by retaining key mathematical information, helping reduce hallucinated substitutions during subsequent reasoning.

3.2 Selecting the Linear Source

We now vary the source assignment while keeping $\alpha = 0.5$ and the attention/low-magnitude selection rule fixed. For each full-attention cache, we compare the third-nearest, second-nearest, and nearest preceding linear layers. As the source approaches the target full-attention layer, ave@4 increases from 46.5% to 47.0% to 47.7% for Qwen, and from 51.2% to 51.8% to 52.5% for Nemotron (Figure 2, right).

We further compare this choice with multi-layer means and position-wise maxima and minima (Appendix B, Table 4). The best aggregated scores, 47.3% for Qwen and 52.1% for Nemotron, both remain 0.4 percentage points below the nearest source. Under this retention rule, the nearest layer performs best among the tested sources and aggregations, supporting its use in StacKV.

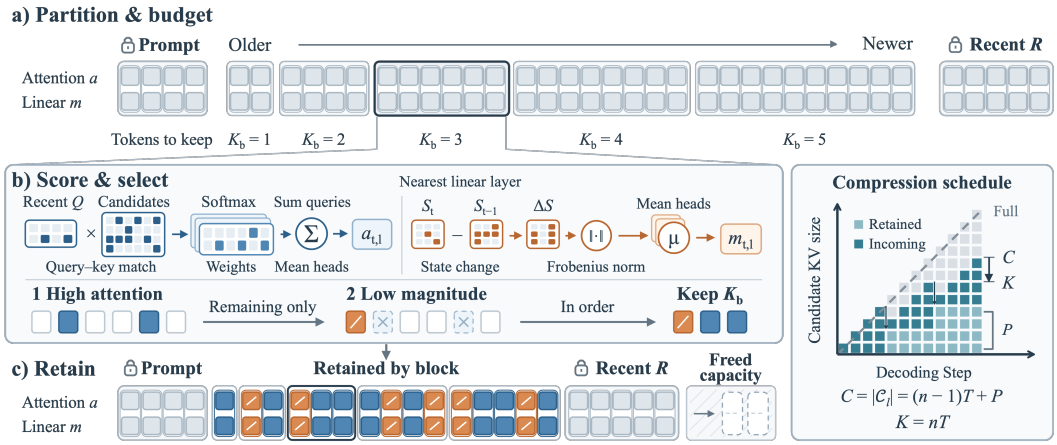

Figure 3: StacKV combines temporal complementarity with spatial coverage. Recorded linear-update scores and recent-query attention jointly select entries within spatial block quotas. The prompt and recent window remain protected; retained KV entries and their linear scores are compacted together.

3.3 Spatial Coverage through Cache Blocks

The selection rule and source assignment determine token rankings but leave positional coverage unconstrained. Global top-k can concentrate retention in some regions and leave others empty. We constrain this allocation through length-proportional quotas over contiguous blocks of the current candidate sequence. Positive quotas prevent whole-block removal in each compression round; smaller blocks enforce coverage at a finer positional scale while maintaining the same target retention density (Appendix D).

Under proportional retention and approximately uniform local spacing, repeated compression makes older history sparser, motivating finer candidate blocks to compensate for the coarser coverage of original token positions (Appendix D). Under the same PolyMath setup and cache budget, we use attention scores alone to compare uniform 128-token blocks with $64/128/256$ and $256/128/64$ schedules over the oldest, middle, and newest thirds of the current candidate sequence.

The $64/128/256$ schedule yields the highest ave@4 among these policies, exceeding global attention-only retention by $0.9/0.8$ percentage points and uniform blocks by $0.5/0.4$ points for Qwen/Nemotron (Table 1). Reversing the same three block widths reduces ave@4 by $1.5/1.0$ points and falls below the global baseline in both models, indicating that the placement of local constraints matters. These attention-only comparisons support finer blocks toward older candidates under the tested setting. Section 4.2 implements this direction with linearly varying block widths, and Section 5.3 evaluates its combination with the linear score.

4 StacKV: Temporal Scoring within Spatial Budgets

StacKV combines two temporal views of each token within a spatially constrained retention set. Linear scores supplement recent-query attention; block quotas determine how many entries survive in each region of the current cache.

4.1 Temporal Complementarity: Hybrid Selection

Recent-query attention saliency. Let $\mathcal{C}_l$ denote the compressible candidate tokens of full-attention layer l, H_l denote the number of attention heads, and let $\mathcal{Q}_l$ contain its R most recent queries. Attention is normalized over candidates only, excluding the protected prompt and recent

window, then aggregated and smoothed:

$$\tilde{a}_{t,l} = \frac{1}{H_l} \sum_{h=1}^{H_l} \sum_{r \in \mathcal{Q}_l} \frac{\exp\left(q_{r,l,h}^\top k_{t,l,h}/\sqrt{d}\right)}{\sum_{u \in \mathcal{C}_l} \exp\left(q_{r,l,h}^\top k_{u,l,h}/\sqrt{d}\right)}, \qquad a_{t,l} = \left[\mathrm{AvgPool}_{\mathcal{K}}(\tilde{a}_{\cdot,l})\right]_t . \tag{2}$$

Here d is the attention head dimension, and $\mathrm{AvgPool}_{\mathcal{K}}$ applies one-dimensional average pooling along the candidate-token order with an odd kernel size $\mathcal{K}$, stride 1, and zero padding of $(\mathcal{K}-1)/2$ on each side. The smoothed score captures each candidate's relative saliency within $\mathcal{C}_l$, accounting for its local neighborhood.

Recorded linear-update signal. Let $\mathcal{P}(l)$ denote the consecutive linear-attention layers immediately preceding full-attention layer l. Motivated by the findings in Section 3.2, we select the nearest source layer and its score as follows, with layer indices increasing in network order:

$$\ell^\star(l) = \max \mathcal{P}(l), \qquad m_{t,l} = u_{t,\ell^\star(l)}, \tag{3}$$

where $u_{t,\ell}$ is the head-averaged state-difference magnitude defined in Equation 1. The score is recorded when token t is processed and stays aligned with its cached KV entry. At compression time, $m_{t,l}$ supplies the historical state-transition view, while $a_{t,l}$ supplies the recent-query view. Following Section 3.1, we select the *lowest* update magnitudes to supplement high attention saliency.

Rank-based hybridization. The two scores measure different quantities, so we combine their rankings through separate budget shares controlled by α. Given a candidate set $\mathcal{C}$ of size $C = |\mathcal{C}|$ and an integer retention budget $0 < K \le C$, we define

$$\begin{aligned} \mathcal{A}_l(\mathcal{C}, K; \alpha) &= \mathrm{Top}^{\max}_{K - \lfloor \alpha K \rfloor}\{a_{t,l} : t \in \mathcal{C}\}, \\ \mathcal{M}_l(\mathcal{C}, K; \alpha) &= \mathrm{Top}^{\min}_{\lfloor \alpha K \rfloor}\{m_{t,l} : t \in \mathcal{C} \setminus \mathcal{A}_l(\mathcal{C}, K; \alpha)\}, \\ \mathcal{H}_l(\mathcal{C}, K; \alpha) &= \mathcal{A}_l(\mathcal{C}, K; \alpha) \cup \mathcal{M}_l(\mathcal{C}, K; \alpha). \end{aligned} \tag{4}$$

where $\mathrm{Top}^{\max}$ and $\mathrm{Top}^{\min}$ return token indices with the largest and smallest scores, respectively. Excluding the attention-selected set from the linear selection prevents duplicates and yields exactly K retained candidates.

4.2 Spatial Coverage: Block-Wise Budgets

We apply the hybrid rule within contiguous blocks so that score ranking operates under local capacity constraints. The spatial coordinate is the current candidate order $[0, C)$ from oldest to newest, where $C = |\mathcal{C}_l|$. Starting at $e_0 = C$, we construct an age-aware partition from newest to oldest. For block b, let e_b be its right boundary and s_b its target width:

$$\begin{aligned} s_b &= \mathrm{round}\Big(s_{\min} + (s_{\max} - s_{\min})\frac{e_b}{C}\Big), \\ e_{b+1} &= \max(0, e_b - s_b), \qquad \mathcal{B}_b = [e_{b+1}, e_b). \end{aligned} \tag{5}$$

Here round returns the nearest integer; $1 \le s_{\min} \le s_{\max}$ are integer parameters controlling the finest and coarsest target granularity, respectively. We stop at $e_B = 0$, where B is the block count. Before rounding, widths vary linearly with e_b/C, giving smaller blocks for older history.

We assign each block a length-proportional target quota r_b. Flooring these quotas leaves D slots to distribute:

$$r_b = K\frac{|\mathcal{B}_b|}{C}, \qquad D = K - \sum_{b=0}^{B-1} \lfloor r_b \rfloor. \tag{6}$$

We floor each r_b and give one extra slot to the D blocks with the largest fractional remainders, yielding integer quotas K_b that sum to K. This rule preserves a common target density K/C across the spatial partition. If the final constructed block (the oldest block) receives zero quota, we merge it with the adjacent second-oldest block and recompute all proportional targets and integer quotas. Our settings satisfy $(K/C)s_{\min} \ge 1$, so every non-final block already has a positive quota, and this merge ensures at least one retained token in every resulting block. Hereafter, B and $\mathcal{B}_b$ refer to the resulting partition. Applying the hybrid rule within each block yields the retained candidate set $\bigcup_{b=0}^{B-1} \mathcal{H}_l(\mathcal{B}_b, K_b; \alpha)$.

4.3 Overall Algorithm

Figure 3 summarizes one compression round. Each full-attention layer maintains its KV cache, aligned linear scores, and a ring buffer of recent queries.

We use an RPC-style schedule (Song et al., 2025): let P be the number of new generated tokens between compression rounds, R the length of the protected observation window, and ρ the retention ratio. Let $T = \mathrm{round}(\rho P)$ be the budget increment per interval. The entire prompt and the newest R tokens remain protected. At the n-th compression round, triggered after $R+nP$ generated tokens, the unprotected cached tokens form the candidate set $\mathcal{C}_l$. Its size C and retention budget K are

$$C = |\mathcal{C}_l| = (n-1)T + P, \qquad K = nT. \tag{7}$$

We partition these candidates, allocate block quotas, and apply hybrid selection within each block to retain exactly K entries. Compacting the KV and recorded-score buffers together preserves their alignment for the next round, while the prompt and recent window remain protected. The complete procedure is given in Algorithm 1 in Appendix E.

5 Experiments

5.1 Experimental Setup

Models and Datasets. We evaluate Qwen3.5-4B with gated delta-rule layers, NVIDIA-Nemotron-Nano-9B-v2 (Nemotron-9B) with Mamba2 recurrent layers, and Kimi-Linear-48B-A3B-Instruct (Kimi) (Team et al., 2025) with Kimi Delta Attention layers. All three models are evaluated on AIME 2024/2025 and HMMT-90 for mathematical reasoning, GPQA-Diamond(Rein et al., 2024) for scientific reasoning, and LiveCodeBench (Jain et al., 2024) for code generation. HMMT-90 combines 90 problems from the 2024, 2025, and 2026 HMMT editions.

Development and Evaluation Protocol. We select α and block parameters $(s_{\min}, s_{\max})$ for Qwen and Nemotron on 500 sampled PolyMath problems using ave@4 at $\rho = 1/4$ (Appendix F), and keep them fixed across evaluation benchmarks. For ave@N, we use the same set of N random seeds across all datasets and compared methods. Kimi reuses Qwen-selected hyperparameters without retuning to test transfer. As a check on model-specific tuning, a shared default configuration also outperforms all four baselines on LiveCodeBench for every model and retention ratio (Appendix F, Table 7).

Hyperparameters. We evaluate five retention levels, with ρ set to $1/16$, $1/8$, $1/4$, $3/8$, or $1/2$, using a compression interval of $P = 2048$ generated tokens and a protected recent-query window of $R = 128$ tokens. The average-pooling kernel size for attention-score smoothing is fixed at $\mathcal{K} = 7$ in all experiments. We use $\alpha = 0.25$ for Qwen and Kimi and $\alpha = 0.5$ for Nemotron, with $(s_{\min}, s_{\max}) = (32, 256)$ and $(64, 256)$, respectively. We use the nearest preceding linear layer as the score source for each full-attention layer.

Baselines. We compare our method with RPC(Song et al., 2025), R-KV(Cai et al., 2026), VaSE(Chang et al., 2026) and LazyEviction(Zhang et al., 2025) at matched retention ratios, using the uncompressed model as a reference. For each baseline, we use the same RPC-style compression schedule as StacKV, and adopt the hyperparameter configuration recommended in its original paper.

Evaluation Setup. All models are evaluated on NVIDIA H20 GPUs using the Nano-vLLM backend, with generation capped at 32,768 new tokens. We evaluate Nemotron in thinking mode and Qwen3.5 in non-thinking mode, as the latter often reaches the officially recommended generation limit when thinking is enabled. We report ave@10 on GPQA-Diamond problems, ave@64 on AIME 2024/2025, ave@16 on HMMT-90, and ave@4 on the first 500 LiveCodeBench problems.

5.2 Experimental Results

Figure 4 compares StacKV with four compression baselines; Appendix G provides full numerical results. StacKV achieves the highest or tied scores across all three models, five benchmarks, and five retention ratios. With Qwen-selected hyperparameters and no Kimi-specific tuning, StacKV strictly leads all four baselines on Kimi in all 25 benchmark–retention settings.

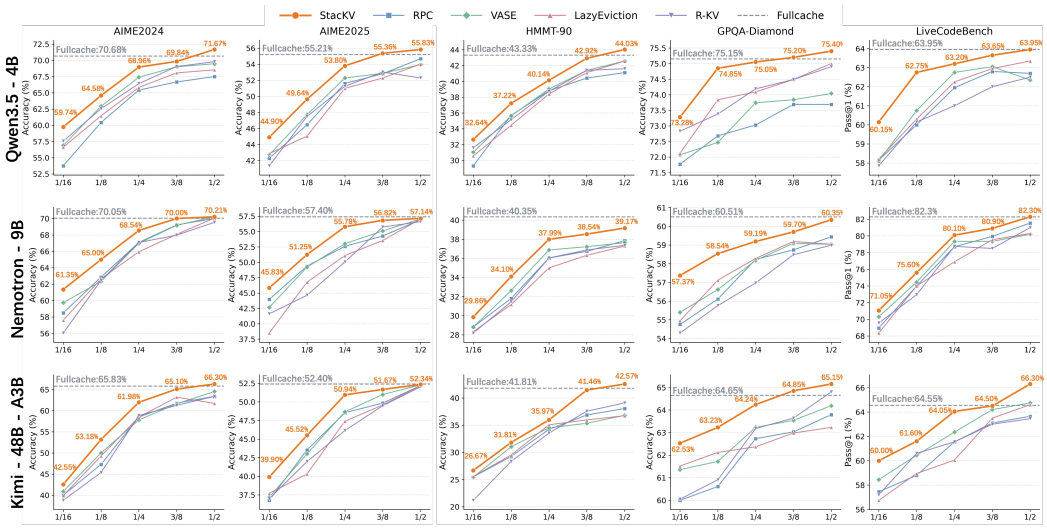

Figure 4: Results on AIME 2024/2025 (ave@64), HMMT-90 (ave@16), GPQA-Diamond (ave@10), and LiveCodeBench (ave@4) at five KV-cache retention ratios. Rows show Qwen, Nemotron, and Kimi; Kimi reuses the Qwen-selected parameters. Dashed lines denote full-cache scores.

Mathematical reasoning. At $\rho = 1/4$, Kimi exceeds the best baseline scores by 3.13 percentage points on AIME 2024 and 2.29 points on AIME 2025 without retuning. Qwen and Nemotron gain 1.56/1.46 points on AIME 2024 and 1.51/2.76 points on AIME 2025, respectively. On HMMT-90 at $\rho = 3/8$, Kimi reaches 41.46%, surpassing the strongest baseline, R-KV (37.57%), by 3.89 points. Thus, the same Qwen-selected configuration improves Kimi's performance across all three mathematical benchmarks, despite the substantial difference in model scale and recurrent-layer design.

General knowledge and reasoning. On GPQA-Diamond, which tests scientific knowledge and reasoning, StacKV improves over the best baselines by 0.86, 0.91, and 1.01 percentage points at $\rho = 1/4$ for Qwen, Nemotron, and Kimi, respectively. At $\rho = 1/16$, Nemotron reaches 57.37%, exceeding the best baseline by 1.97 points. These results indicate that the advantage is not confined to structured mathematical traces: it also persists on questions requiring broad factual knowledge. The larger Nemotron margin at the most aggressive budget further shows that StacKV remains effective when the available KV capacity is sharply constrained.

Code generation. On LiveCodeBench at $\rho = 1/4$, StacKV reaches 63.20%, 80.10%, and 64.05% for Qwen, Nemotron, and Kimi, improving over the best baselines by 0.45, 0.75, and 1.70 percentage points, respectively. Kimi's 1.70-point gain is the largest among the three models at this budget, showing that the transferred configuration remains effective beyond mathematical and scientific reasoning.

5.3 Ablating Temporal and Spatial Components

Table 2 isolates low-magnitude selection and age-aware blocks under Section 5.1's fixed settings. Low-Mag uses the nearest preceding linear layer and each model's selected α; blocks follow Equation 5. Variants without blocks select over the full candidate set.

Table 2: Component ablation on AIME 2024 at $\rho = 1/4$ (ave@64, %). Attn denotes attention; bold marks the best score per model.

Variant	Qwen3.5-4B	Nemotron-9B	Kimi
Attn only	65.42	67.08	58.49
Attn + Low-Mag	67.66	68.02	60.10
Attn + blocks	67.92	67.86	59.48
Full method	**68.96**	**68.54**	**61.98**

Each component improves on attention-only retention. Low-Mag adds 2.24, 0.94, and 1.61 percentage points for Qwen, Nemotron, and Kimi, respectively; blocks add 2.50, 0.78, and

Table 3: Qwen3.5-4B efficiency across KV-cache retention budgets. Throughput is reported in tokens/s; RPC uses the same batch size as the corresponding row. Speedups compare StacKV with each backend's FullCache baseline; bold marks the peak speedup.

Retention ρ	Hugging Face (HF)				Nano-vLLM			
	Batch size	RPC	StacKV	Speedup	Batch size	RPC	StacKV	Speedup
FullCache	48	556.52	556.52	$1.00\times$	77	2749.63	2749.63	$1.00\times$
50%	64	912.29	914.41	$1.64\times$	128	3766.64	3707.96	$1.35\times$
37.5%	80	1036.69	1040.61	$1.87\times$	160	4139.38	4077.47	$1.48\times$
25%	96	1131.39	1128.93	$2.03\times$	192	4719.60	4643.47	$1.69\times$
12.5%	160	1873.07	1886.60	$3.39\times$	320	5437.93	5349.08	$1.95\times$
6.25%	192	2150.23	2168.05	$\mathbf{3.90}\times$	418	5700.18	5656.05	$\mathbf{2.06}\times$

0.99 points. The temporal supplement therefore helps under global selection, while spatial coverage helps with attention scores alone.

Combining both improves over attention alone by 3.54, 1.46, and 3.49 points and over the stronger single component by 1.04, 0.52, and 1.88 points, respectively. Both components benefit Kimi under the transferred Qwen configuration.

5.4 Efficiency Analysis

Throughput. Table 3 evaluates Qwen3.5-4B with a maximum generation length of 32,768 tokens, using the largest feasible batch size for each configuration to measure throughput. At the reported batch sizes, StacKV achieves 1.64–$3.90\times$ speedups in Hugging Face (HF) and 1.35–$2.06\times$ in Nano-vLLM over FullCache. Throughput peaks at 6.25% retention: 2168.05 tokens/s in HF and 5656.05 tokens/s in Nano-vLLM.

Kernel overhead. We compare our method with RPC at the same retention budget and batch size to assess the end-to-end overhead of our additional kernels. In HF, throughput differs by at most 0.83%, indicating negligible performance impact. In Nano-vLLM, throughput decreases by 0.77%–1.63%, averaging 1.41% across the five retention budgets, while retaining the speedups over FullCache.

Memory. With a fixed batch size of 48, 128 input tokens, and 32,768 generated tokens, StacKV at $\rho = 1/4$ reaches a 24.64 GiB whole-device peak and adds 44.28 MiB of compression overhead in HF. Across all retention levels, this overhead exceeds RPC by at most 1.80 MiB, while the whole-device peaks differ by at most 0.15 GiB (Appendix H).

Efficient score computation. In Nano-vLLM, custom Triton kernels compute full state-change scores only for selected source layers. Backend-specific prefill recurrences and fused decode scoring reuse state tiles to include decay and writing. Aligned FP32 token scores are reused by batched blockwise selection, avoiding per-token state-history storage and score recomputation.

6 Conclusion

StacKV couples temporal complementarity with spatial coverage for hybrid KV-cache compression. Recorded low-magnitude recurrent updates supplement recent-query attention, while block-wise quotas distribute retained entries across cache positions with finer coverage for older candidates. Experiments across three hybrid models show improved reasoning and code-generation accuracy at matched cache budgets, with both components contributing in controlled ablations. The Qwen-selected parameters transfer directly to Kimi without retuning, while Qwen throughput measurements demonstrate acceleration with little overhead relative to RPC.

AI USE STATEMENT

Generative AI tools were used to assist with software implementation and to polish the manuscript for clarity and readability. All AI-assisted code and text were reviewed and verified by the authors. Generative AI tools were not used to generate datasets or experimental results. The authors take full responsibility for the final content of this work.

ETHICS STATEMENT

This work studies KV-cache compression for language-model inference and does not involve human subjects or the collection of personal or sensitive data. The experiments use publicly released models and established benchmarks. The proposed technique may reduce the computational cost of language-model inference but, like other efficiency techniques, may also facilitate harmful downstream applications of the underlying models.

REPRODUCIBILITY STATEMENT

Section 5.1 reports the models, benchmarks, evaluation protocol, hyperparameters, and hardware setup. The method and complete algorithm are described in Section 4 and Appendix E. The appendix further provides the hyperparameter search, full numerical results, and memory measurements.

REFERENCES

Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Zou, Atri Rudra, and Christopher Ré. Zoology: Measuring and improving recall in efficient language models, 2023. URL https://arxiv.org/abs/2312.04927.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling, 2025. URL https://arxiv.org/abs/2406.02069.

Zefan Cai, Wen Xiao, Hanshi Sun, Cheng Luo, Yikai Zhang, Ke Wan, Yucheng Li, Yeyang Zhou, Li-Wen Chang, Jiuxiang Gu, Zhen Dong, Anima Anandkumar, Abedelkadir Asi, and Junjie Hu. R-kv: Redundancy-aware kv cache compression for reasoning models, 2026. URL https://arxiv.org/abs/2505.24133.

Ting-Yun Chang, Harvey Yiyun Fu, Deqing Fu, Chenghao Yang, Jesse Thomason, and Robin Jia. Value-aware stochastic kv cache eviction for reasoning models, 2026. URL https://arxiv.org/abs/2606.03928.

Pengyu Cheng, Jiacheng Wang, Tianle Chen, Bei Liu, Xiaofeng Hou, and Jiacheng Liu. Desirekv: Decoupling sensitivity and importance for reasoning-aware KV cache compression. In Sven Koenig, Chad Jenkins, and Matthew E. Taylor (eds.), *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2026, Singapore, January 20-27, 2026*, pp. 20518–20526. AAAI Press, 2026. doi: 10.1609/AAAI.V40I25.39187. URL https://doi.org/10.1609/aaai.v40i25.39187.

Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality, 2024. URL https://arxiv.org/abs/2405.21060.

Soham De, Samuel L. Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, Guillaume Desjardins, Arnaud Doucet, David Budden, Yee Whye Teh, Razvan Pascanu, Nando De Freitas, and Caglar Gulcehre. Griffin: Mixing gated linear recurrences with local attention for efficient language models, 2024. URL https://arxiv.org/abs/2402.19427.

Xin Dong, Yonggan Fu, Shizhe Diao, Wonmin Byeon, Zijia Chen, Ameya Sunil Mahabaleshwarkar, Shih-Yang Liu, Matthijs Van Keirsbilck, Min-Hung Chen, Yoshi Suhara, Yingyan Lin, Jan Kautz, and Pavlo Molchanov. Hymba: A hybrid-head architecture for small language models, 2024. URL https://arxiv.org/abs/2411.13676.

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, 2025. URL https://arxiv.org/abs/2407.11550.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2024. URL https://arxiv.org/abs/2312.00752.

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, 2025. URL https://arxiv.org/abs/2401.18079.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code, 2024. URL https://arxiv.org/abs/2403.07974.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5156–5165. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/katharopoulos20a.html.

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, 2023. URL https://arxiv.org/abs/2309.06180.

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, 2024. URL https://arxiv.org/abs/2404.14469.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, Omri Abend, Raz Alon, Tomer Asida, Amir Bergman, Roman Glozman, Michael Gokhman, Avashalom Manevich, Nir Ratner, Noam Rozen, Erez Shwartz, Mor Zusman, and Yoav Shoham. Jamba: A hybrid transformer-mamba language model, 2024. URL https://arxiv.org/abs/2403.19887.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

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 Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32332–32344. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/liu24bz.html.

NVIDIA, :, Aarti Basant, Abhijit Khairnar, Abhijit Paithankar, Abhinav Khattar, Adithya Renduchintala, Aditya Malte, Akhiad Bercovich, Akshay Hazare, Alejandra Rico, Aleksander Ficek, Alex Kondratenko, Alex Shaposhnikov, Alexander Bukharin, Ali Taghibakhshi, Amelia Barton, Ameya Sunil Mahabaleshwarkar, Amy Shen, Andrew Tao, Ann Guan, Anna Shors, Anubhav Mandarwal, Arham Mehta, Arun Venkatesan, Ashton Sharabiani, Ashwath Aithal, Ashwin Poojary, Ayush Dattagupta, Balaram Buddharaju, Banghua Zhu, Barnaby Simkin, Bilal Kartal, Bita Darvish Rouhani, Bobby Chen, Boris Ginsburg, Brandon Norick, Brian Yu, Bryan Catanzaro, Charles Wang, Charlie Truong, Chetan Mungekar, Chintan Patel, Chris Alexiuk, Christian Munley, Christopher Parisien, Dan Su, Daniel Afrimi, Daniel Korzekwa, Daniel Rohrer, Daria Gitman, David Mosallanezhad, Deepak Narayanan, Dima Rekesh, Dina Yared, Dmytro Pykhtar, Dong Ahn, Duncan Riach, Eileen Long, Elliott Ning, Eric Chung, Erick Galinkin, Evelina Bakhturina, Gargi Prasad, Gerald Shen, Haifeng Qian, Haim Elisha, Harsh Sharma, Hayley Ross, Helen Ngo, Herman Sahota, Hexin Wang, Hoo Chang Shin, Hua Huang, Iain Cunningham, Igor Gitman, Ivan Moshkov, Jaehun Jung, Jan Kautz, Jane Polak Scowcroft, Jared Casper, Jian Zhang, Jiaqi Zeng, Jimmy Zhang, Jinze Xue, Jocelyn Huang, Joey Conway, John Kamalu, Jonathan

Cohen, Joseph Jennings, Julien Veron Vialard, Junkeun Yi, Jupinder Parmar, Kari Briski, Katherine Cheung, Katherine Luna, Keith Wyss, Keshav Santhanam, Kezhi Kong, Krzysztof Pawelec, Kumar Anik, Kunlun Li, Kushan Ahmadian, Lawrence McAfee, Laya Sleiman, Leon Derczynski, Luis Vega, Maer Rodrigues de Melo, Makesh Narsimhan Sreedhar, Marcin Chochowski, Mark Cai, Markus Kliegl, Marta Stepniewska-Dziubinska, Matvei Novikov, Mehrzad Samadi, Meredith Price, Meriem Boubdir, Michael Boone, Michael Evans, Michal Bien, Michal Zawalski, Miguel Martinez, Mike Chrzanowski, Mohammad Shoeybi, Mostofa Patwary, Namit Dhameja, Nave Assaf, Negar Habibi, Nidhi Bhatia, Nikki Pope, Nima Tajbakhsh, Nirmal Kumar Juluru, Oleg Rybakov, Oleksii Hrinchuk, Oleksii Kuchaiev, Oluwatobi Olabiyi, Pablo Ribalta, Padmavathy Subramanian, Parth Chadha, Pavlo Molchanov, Peter Dykas, Peter Jin, Piotr Bialecki, Piotr Januszewski, Pradeep Thalasta, Prashant Gaikwad, Prasoon Varshney, Pritam Gundecha, Przemek Tredak, Rabeeh Karimi Mahabadi, Rajen Patel, Ran El-Yaniv, Ranjit Rajan, Ria Cheruvu, Rima Shahbazyan, Ritika Borkar, Ritu Gala, Roger Waleffe, Ruoxi Zhang, Russell J. Hewett, Ryan Prenger, Sahil Jain, Samuel Kriman, Sanjeev Satheesh, Saori Kaji, Sarah Yurick, Saurav Muralidharan, Sean Narenthiran, Seonmyeong Bak, Sepehr Sameni, Seungju Han, Shanmugam Ramasamy, Shaona Ghosh, Sharath Turuvekere Sreenivas, Shelby Thomas, Shizhe Diao, Shreya Gopal, Shrimai Prabhumoye, Shubham Toshniwal, Shuoyang Ding, Siddharth Singh, Siddhartha Jain, Somshubra Majumdar, Soumye Singhal, Stefania Alborghetti, Syeda Nahida Akter, Terry Kong, Tim Moon, Tomasz Hliwiak, Tomer Asida, Tony Wang, Tugrul Konuk, Twinkle Vashishth, Tyler Poon, Udi Karpas, Vahid Noroozi, Venkat Srinivasan, Vijay Korthikanti, Vikram Fugro, Vineeth Kalluru, Vitaly Kurin, Vitaly Lavrukhin, Wasi Uddin Ahmad, Wei Du, Wonmin Byeon, Ximing Lu, Xin Dong, Yashaswi Karnati, Yejin Choi, Yian Zhang, Ying Lin, Yonggan Fu, Yoshi Suhara, Zhen Dong, Zhiyu Li, Zhongbo Zhu, and Zijia Chen. Nvidia nemotron nano 2: An accurate and efficient hybrid mamba-transformer reasoning model, 2025. URL https://arxiv.org/abs/2508.14444.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Anselm Levskaya, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference, 2022. URL https://arxiv.org/abs/2211.05102.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

Liliang Ren, Yang Liu, Yadong Lu, Yelong Shen, Chen Liang, and Weizhu Chen. Samba: Simple hybrid state space models for efficient unlimited context language modeling, 2025. URL https://arxiv.org/abs/2406.07522.

Jiwon Song, Dongwon Jo, Yulhwa Kim, and Jae-Joon Kim. Reasoning path compression: Compressing generation trajectories for efficient llm reasoning, 2025. URL https://arxiv.org/abs/2505.13866.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models, 2023. URL https://arxiv.org/abs/2307.08621.

Hanlin Tang, Yang Lin, Jing Lin, Qingsen Han, Shikuan Hong, Yiwu Yao, and Gongyi Wang. Razorattention: Efficient kv cache compression through retrieval heads, 2024a. URL https://arxiv.org/abs/2407.15891.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context llm inference, 2024b. URL https://arxiv.org/abs/2406.10774.

Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, Wentao Li, Enzhe Lu, Weizhou Liu, Yanru Chen, Weixin Xu, Longhui Yu, Yejie Wang, Yu Fan, Longguang Zhong, Enming Yuan, Dehao Zhang, Yizhi Zhang, T. Y. Liu, Haiming Wang, Shengjun Fang, Weiran He, Shaowei Liu, Yiwei Li, Jianlin Su, Jiezhong Qiu, Bo Pang, Junjie Yan, Zhejun Jiang, Weixiao Huang, Bohong Yin, Jiacheng You, Chu Wei, Zhengtao Wang, Chao Hong, Yutian Chen, Guanduo Chen, Yucheng Wang, Huabin Zheng, Feng

Wang, Yibo Liu, Mengnan Dong, Zheng Zhang, Siyuan Pan, Wenhao Wu, Yuhao Wu, Longyu Guan, Jiawen Tao, Guohong Fu, Xinran Xu, Yuzhi Wang, Guokun Lai, Yuxin Wu, Xinyu Zhou, Zhilin Yang, and Yulun Du. Kimi linear: An expressive, efficient attention architecture, 2025. URL https://arxiv.org/abs/2510.26692.

Llama Team. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024. doi: 10.48550/ARXIV.2407.21783. URL https://doi.org/10.48550/arXiv.2407.21783.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Yiming Wang, Pei Zhang, Jialong Tang, Haoran Wei, Baosong Yang, Rui Wang, Chenshu Sun, Feitong Sun, Jiran Zhang, Junxuan Wu, Qiqian Cang, Yichang Zhang, Fei Huang, Junyang Lin, Fei Huang, and Jingren Zhou. Polymath: Evaluating mathematical reasoning in multilingual contexts, 2025. URL https://arxiv.org/abs/2504.18428.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks, 2024. URL https://arxiv.org/abs/2309.17453.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025a. URL https://arxiv.org/abs/2505.09388.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training, 2024. URL https://arxiv.org/abs/2312.06635.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule, 2025b. URL https://arxiv.org/abs/2412.06464.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length, 2025c. URL https://arxiv.org/abs/2406.06484.

Haoyue Zhang, Hualei Zhang, Xiaosong Ma, Jie Zhang, and Song Guo. Lazyeviction: Lagged kv eviction with attention pattern observation for efficient long reasoning, 2025. URL https://arxiv.org/abs/2506.15969.

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. H_2o: Heavy-hitter oracle for efficient generative inference of large language models, 2023. URL https://arxiv.org/abs/2306.14048.

A State-Update Formulas

For gated delta-rule layers, let $\widetilde{S}_t = e^{g_t} S_{t-1}$ denote the decayed state, with layer and head indices suppressed for clarity. The state difference is

$$S_t - S_{t-1} = (e^{g_t} - 1)S_{t-1} + \beta_t k_t \left(v_t - \widetilde{S}_t^\top k_t\right)^\top, \tag{8}$$

where e^{g_t} and β_t control decay and writing, k_t is the normalized key, and v_t is the value vector. For Mamba2 layers,

$$S_t - S_{t-1} = \left(e^{\Delta_t a_h} - 1\right) S_{t-1} + \Delta_t x_t B_t^\top, \tag{9}$$

where $\Delta_t > 0$ is the input-dependent time step, $a_h < 0$ is the learned head-specific decay parameter, and x_t and B_t are the input and state-write vectors.

For Kimi Delta Attention (KDA) layers in Kimi Linear, the decay varies across key channels. Let $S_t \in \mathbb{R}^{d_k \times d_v}$, where d_k and d_v are the key and value dimensions, and let $g_t \in \mathbb{R}^{d_k}$ contain the channel-wise log-decay values. Define $D_t = \mathrm{diag}(\exp(g_t))$ and $\widetilde{S}_t = D_t S_{t-1}$, with the exponential applied elementwise. The recurrence and its corresponding state difference are

$$S_t - S_{t-1} = (D_t - I)S_{t-1} + \beta_t k_t \left(v_t - \widetilde{S}_t^\top k_t\right)^\top, \tag{10}$$

where I is the $d_k \times d_k$ identity, k_t is the normalized key, and β_t controls writing. Setting $D_t = e^{g_t} I$ with a shared scalar log-decay recovers Equation 8. In all three cases, Equation 1 measures the complete state change, including decay and writing.

B Source-Layer Selection and Aggregation

To further examine the source-layer choice in Section 3.2, we compare single-layer sources with aggregations of the three nearest preceding linear layers. The comparison follows the same PolyMath setup: 500 sampled problems, ave@4, a retention ratio of $\rho = 1/4$, and a fixed $\alpha = 0.5$. Only the source-layer selection or aggregation rule varies; all other settings are held fixed.

Let L_1, L_2, and L_3 denote the nearest, second-nearest, and third-nearest preceding linear layers, respectively, and let $u_i = u_{t,L_i}$ be the update magnitude from Equation 1 for the same candidate token t. Each rule in Table 4 produces a scalar score per token, whose lowest values supply the low-magnitude selection. Means, maxima, and minima are computed across source layers at each token position.

Table 4: Source-layer selection and aggregation on PolyMath at $\rho = 0.25$. Model columns report ave@4 (%); bold marks the best score for each model.

Aggregation rule	Token score	Qwen3.5-4B	Nemotron-9B
Three-layer mean	$(u_1 + u_2 + u_3)/3$	47.2	51.6
L_1 only (nearest)	u_1	**47.7**	**52.5**
L_2 only	u_2	47.0	51.8
L_3 only	u_3	46.5	51.2
Mean of L_1, L_2	$(u_1 + u_2)/2$	47.3	52.0
Mean of L_1, L_3	$(u_1 + u_3)/2$	46.5	51.5
Mean of L_2, L_3	$(u_2 + u_3)/2$	45.9	51.4
Position-wise maximum	$\max(u_1, u_2, u_3)$	47.1	51.3
Position-wise minimum	$\min(u_1, u_2, u_3)$	46.2	52.1

C Formula-Like Tokens in Score-Based Selections

We compare the token types selected by low-magnitude linear-update scores (Low-Mag) and attention scores (Attention/RPC) at a retention ratio of $\rho = 1/4$ (25%). The two rules independently select tokens from the same candidate history under identical retention budgets; Low-Mag does not exclude attention-selected tokens. We define *formula-like tokens* as numbers, equality and other mathematical symbols, and LaTeX math commands such as `\frac` and `\sqrt`. For each selection rule, we report the percentage of selected tokens that are formula-like, with all selected tokens as the denominator.

Table 5: Formula-like tokens as a percentage of selected tokens at 25% retention. The difference is Low-Mag minus Attention/RPC, in percentage points (pp).

Model	Dataset	Low-Mag	Attention/RPC	Difference (pp)
Qwen3.5-4B	AIME 2024	91.89%	72.52%	+19.37
	GPQA	84.18%	67.06%	+17.12
Nemotron Nano 9B v2	AIME 2024	89.98%	68.26%	+21.72
	GPQA	80.98%	58.37%	+22.61

Low-Mag yields a higher formula-like token share in all four settings, with increases of 17.12–22.61 percentage points. This consistent preference for mathematical content helps explain how the linear signal complements attention by preserving details useful for subsequent reasoning.

D Spatial Coverage and the Rationale for Finer Older Blocks

At round r, let C_r be the candidate count and $k_r > 0$ the retention budget. Index the B resulting blocks after the possible merge in Section 4.2 in candidate order. Block b contains $n_{b,r}$ candidates and retains a feasible positive quota $q_{b,r}$, with $\sum_b n_{b,r} = C_r$ and $\sum_b q_{b,r} = k_r$. Equation 7 gives the current-candidate retention density

$$\lambda_r = \frac{k_r}{C_r} = \frac{rT}{(r-1)T + P}, \qquad T = \mathrm{round}(\rho P), \tag{11}$$

where $r \geq 1$, $P > 0$, and $0 < T \leq P$; λ_r generally differs from ρ.

Single-round coverage. A global ranking can empty block b if and only if $k_r \leq C_r - n_{b,r}$, since all retained entries must then fit outside it. With positive block quotas, let $h_{b,r} = n_{b,r} - q_{b,r}$ be the number of deletions, and let $g_{b,r}$ and G_r be the longest deletion runs within block b and across the entire candidate sequence, respectively. Largest-remainder allocation gives

$$g_{b,r} \leq h_{b,r} \leq n_{b,r} - \lfloor \lambda_r n_{b,r} \rfloor = \lceil (1 - \lambda_r) n_{b,r} \rceil, \tag{12}$$

$$G_r \leq \overline{G}_r := \begin{cases} h_{1,r}, & B = 1, \\ \max\limits_{1 \leq b < B} (h_{b,r} + h_{b+1,r}), & B \geq 2. \end{cases} \tag{13}$$

Proof. Block b has exactly $h_{b,r}$ deleted entries, and $q_{b,r} \geq \lfloor \lambda_r n_{b,r} \rfloor$ gives Equation 12. Since every block retains at least one entry, a deletion run can intersect at most two adjacent blocks, proving Equation 13. Consequently, any interval of original token positions containing more than $\overline{G}_r$ current candidates retains at least one of them. At fixed λ_r, smaller neighboring blocks tighten the corresponding density-based bounds on deletion runs.

Age and positional resolution. An already eligible token cohort can only lose candidates over subsequent rounds. To relate this loss to age, consider an idealized model of proportional retention: each cohort retains a fraction λ_j at round j, ignoring quota rounding and blocks crossing cohort boundaries. For a cohort first eligible in round c, its remaining density relative to its original token count before round $r \geq c$ is

$$\eta_{c,r} = \prod_{j=c}^{r-1} \lambda_j, \qquad \eta_{r,r} = 1. \tag{14}$$

All factors lie in $(0, 1]$, so earlier cohorts have no larger density under this model. Approximating local candidate spacing by $1/\eta_{c,r}$, a block of $s_{c,r}$ candidates represents an original-position scale $L_{c,r}$; matching a target scale $L_\star$ calls for

$$L_{c,r} \approx \frac{s_{c,r}}{\eta_{c,r}}, \qquad s_{c,r} \approx \eta_{c,r} L_\star, \tag{15}$$

subject to feasible positive quotas. Thus, finer candidate blocks compensate for the lower density of older history while keeping the target retention fraction unchanged. Section 4.2 uses candidate order

as an age proxy and a bounded linear schedule to implement this direction. The approximation motivates that direction; it does not establish an optimal schedule or an accuracy guarantee. The exact coverage bounds remain single-round guarantees in current-candidate order, without a cumulative gap bound on the original token timeline.

E Overall StacKV Algorithm

Algorithm 1 combines the temporal scores, spatial budgets, and compression schedule in Section 4. Candidate indices follow the current cache order from oldest to newest; retained entries preserve their original token positions. We assume $0 < T \leq P$ and $(K/C)s_{\min} \geq 1$ at each compression round, as in Section 4.2; selecting zero entries returns the empty set.

Algorithm 1 StacKV: online scoring and periodic KV-cache compression

Require: Hybrid model and prompt; interval P, recent window R, retention ratio ρ; parameter α, block widths $s_{\min}, s_{\max}$, pooling kernel $\mathcal{K}$.
Ensure: Generated continuation and compressed KV caches at full-attention layers.

$T \leftarrow \mathrm{round}(\rho P)$; $n \leftarrow 1$.
For each full-attention layer l, set $\ell^\star(l) \leftarrow \max \mathcal{P}(l)$.
Prefill the prompt; initialize aligned score buffers and recent-query ring buffers.
for each generated-token step $g = 1, 2, \ldots$ until generation stops **do**
  Process the next token t; record all source-layer scores using Equation 1.
  for each full-attention layer l **do**
    Append the KV entry and its score $m_{t,l} \leftarrow u_{t,\ell^\star(l)}$.
    Update $\mathcal{Q}_l$ with the new query, keeping the most recent R queries.
  end for
  if $g = R + nP$ **then**
    for each full-attention layer l **do**
      $\mathcal{C}_l \leftarrow$ cached tokens excluding the prompt and newest R tokens.
      $C \leftarrow |\mathcal{C}_l| = (n-1)T + P$; $K \leftarrow nT$.
      Compute $a_{t,l}$ for all $t \in \mathcal{C}_l$ using Equation 2.
      Build blocks $\{\mathcal{B}_b\}_{b=0}^{B-1}$ from newest to oldest using Equation 5.
      $\{K_b\} \leftarrow \textsc{AllocateQuotas}(\{\mathcal{B}_b\}, C, K)$.
      if $K_{B-1} = 0$ **then**
        Merge the oldest block $\mathcal{B}_{B-1}$ into $\mathcal{B}_{B-2}$; $B \leftarrow B - 1$.
        $\{K_b\} \leftarrow \textsc{AllocateQuotas}(\{\mathcal{B}_b\}, C, K)$.
      end if
      $\mathcal{I}_l \leftarrow \emptyset$.
      for $b = 0, \ldots, B-1$ **do**
        $k_a \leftarrow K_b - \lfloor \alpha K_b \rfloor$.
        $\mathcal{A} \leftarrow \mathrm{Top}^{\max}_{k_a}\{a_{t,l} : t \in \mathcal{B}_b\}$.
        $\mathcal{M} \leftarrow \mathrm{Top}^{\min}_{K_b - k_a}\{m_{t,l} : t \in \mathcal{B}_b \setminus \mathcal{A}\}$.
        $\mathcal{I}_l \leftarrow \mathcal{I}_l \cup \mathcal{A} \cup \mathcal{M}$.
      end for
      Retain the prompt, $\mathcal{I}_l$, and newest R tokens in chronological order.
      Compact KV entries, recorded scores, and token positions with the same indices.
    end for
    $n \leftarrow n + 1$.
  end if
end for

function $\textsc{AllocateQuotas}(\{\mathcal{B}_b\}, C, K)$
  $r_b \leftarrow K|\mathcal{B}_b|/C$; $K_b \leftarrow \lfloor r_b \rfloor$ for each block b.
  $D \leftarrow K - \sum_b K_b$.
  Add one slot to each of the D blocks with the largest $r_b - \lfloor r_b \rfloor$.
  return $\{K_b\}$.
end function

F HYPERPARAMETER SELECTION, SENSITIVITY, AND SHARED DEFAULTS

We examine how α and the block granularity affect performance. The results support the configurations used in Section 5.1.

Development configuration. Following the setup in Section 3, we evaluate both models on 500 problems sampled from PolyMath using ave@4 at a retention ratio of $\rho = 1/4$ (25%). We use the compression method described in Section 4.

Search space. We sweep $\alpha \in \{0, 0.25, 0.5, 0.75, 1\}$. For each α, we sweep $s_{\min} \in \{16, 32, 64\}$ and $s_{\max} \in \{128, 256, 512\}$, keeping the other settings fixed. Both $\alpha = 0$ and $\alpha = 1$ retain block-wise budgets. Table 6 reports the mean over nine block settings for each α; Figure 5 shows $\alpha = 0.25$ for Qwen and $\alpha = 0.5$ for Nemotron.

Signal allocation. Mean scores peak at $\alpha = 0.25$ for Qwen (47.64%) and $\alpha = 0.5$ for Nemotron (52.48%) (Table 6). These settings exceed attention-only selection by 0.93/0.95 percentage points and linear-only selection by 1.45/1.04 points for Qwen/Nemotron. All three intermediate values of α outperform both single-signal endpoints on average, supporting complementary signals within the block-wise policy. These results favor $\alpha = 0.25$ for Qwen and $\alpha = 0.5$ for Nemotron.

Table 6: Sensitivity to α on the PolyMath development subset. Each entry is the mean ave@4 (%) over nine block settings at $\rho = 1/4$. Bold marks the best α for each model.

α	Qwen3.5-4B	Nemotron-9B
0	46.71	51.53
0.25	**47.64**	51.92
0.5	47.33	**52.48**
0.75	46.84	52.03
1	46.19	51.44

Block granularity. At each model's selected α, Qwen3.5-4B reaches its highest score of 48.3% at $(s_{\min}, s_{\max}) = (32, 256)$, while Nemotron-9B peaks at 53.1% with $(64, 256)$ (Figure 5). Scores span 47.1–48.3% for Qwen and 51.6–53.1% for Nemotron across the grid. Increasing $s_{\max}$ from 128 to 256 improves every row for both models, whereas a further increase to 512 gives no consistent benefit. The shared preference for $s_{\max} = 256$ and different best values of $s_{\min}$ support using model-specific block granularity. The best settings match those used in the main experiments.

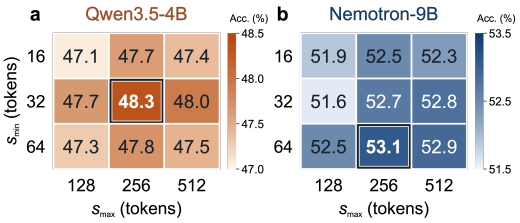

Figure 5: PolyMath ave@4 (%) across block settings at $\rho = 1/4$, using $\alpha = 0.25$ for Qwen and $\alpha = 0.5$ for Nemotron. Outlined cells mark the best setting for each model.

Shared default configuration. To assess whether the gains require model-specific hyperparameters, we evaluate StacKV on LiveCodeBench with a single default configuration, $\alpha = 0.5$ and $(s_{\min}, s_{\max}) = (32, 256)$, shared across all three models and all five retention ratios. LiveCodeBench provides a demanding comparison because the gains in the main experiments are relatively modest on this benchmark. Table 7 compares these results with the four baselines reported in Appendix G at matched retention ratios. With shared defaults, StacKV exceeds the strongest baseline in all 15 model–retention settings by 0.45–1.90 percentage points, showing that its advantage persists without model-specific choices of these hyperparameters.

Table 7: LiveCodeBench ave@4 (%) with shared default hyperparameters. StacKV uses $\alpha = 0.5$, $s_{\min} = 32$, and $s_{\max} = 256$ for every model and retention ratio. Baseline scores are reproduced from Tables 8–10. Ave. is the arithmetic mean over the five retention ratios; bold marks the best score per model and column.

Method	Retention ratio ρ					Ave.
	$1/16$	$1/8$	$1/4$	$3/8$	$1/2$	
Qwen3.5-4B						
RPC	58.15	60.00	61.95	62.80	62.70	61.12
R-KV	57.85	60.15	61.00	62.00	62.50	60.70
VaSE	58.10	60.75	62.75	63.05	62.35	61.40
LazyEviction	58.20	60.30	62.25	62.95	63.35	61.41
StacKV (shared defaults)	**59.75**	**62.00**	**63.20**	**63.55**	**63.80**	**62.46**
Nemotron-9B						
RPC	68.95	74.00	78.70	79.95	81.55	76.63
R-KV	69.55	72.95	78.75	78.50	81.00	76.15
VaSE	70.30	74.45	79.35	79.35	80.20	76.73
LazyEviction	68.35	74.00	76.90	79.55	80.30	75.82
StacKV (shared defaults)	**71.30**	**75.70**	**80.00**	**81.10**	**82.40**	**78.10**
Kimi						
RPC	57.45	58.80	61.55	63.10	63.65	60.91
R-KV	57.20	60.60	61.55	63.00	63.45	61.16
VaSE	58.45	60.45	62.35	64.20	64.75	62.04
LazyEviction	56.75	58.95	60.05	63.50	64.60	60.77
StacKV (shared defaults)	**59.75**	**61.75**	**64.25**	**64.80**	**66.30**	**63.37**

G Full Experimental Results

Tables 8–10 report all scores underlying Figure 4. All values are percentages, using ave@64 on AIME 2024/2025, ave@16 on HMMT-90, ave@10 on GPQA-Diamond, and ave@4 on LiveCodeBench, as defined in Section 5.1. Ave. reports the arithmetic mean over the five retention ratios. Full-cache scores appear beneath each benchmark name; bold marks the best compressed-method score in each column, including ties. Kimi uses the Qwen-selected configuration without retuning: $\alpha = 0.25$ and $(s_{\min}, s_{\max}) = (32, 256)$, with $P = 2048$, $R = 128$, $\mathcal{K} = 7$, and the nearest preceding linear layer as the score source.

Table 8: Full results for Qwen3.5-4B at five retention ratios ρ.

Benchmark	Method	$1/16$	$1/8$	$1/4$	$3/8$	$1/2$	Ave.
AIME 2024 Full cache: 70.68	RPC	53.75	60.42	65.42	66.67	67.50	62.75
	R-KV	57.55	62.55	66.41	69.06	69.79	65.07
	VaSE	56.88	62.92	67.40	69.01	69.48	65.14
	LazyEviction	56.61	61.46	65.83	68.07	68.54	64.10
	StacKV (ours)	**59.74**	**64.58**	**68.96**	**69.84**	**71.67**	**66.96**
AIME 2025 Full cache: 55.21	RPC	42.29	46.46	51.61	52.81	54.69	49.57
	R-KV	41.35	47.50	51.25	53.02	52.29	49.08
	VaSE	42.71	47.71	52.29	52.86	53.96	49.91
	LazyEviction	42.86	45.05	50.99	52.29	54.01	49.04
	StacKV (ours)	**44.90**	**49.64**	**53.80**	**55.36**	**55.83**	**51.91**
HMMT-90 Full cache: 43.33	RPC	29.31	35.69	38.82	40.42	41.11	37.07
	R-KV	31.59	35.14	38.75	41.39	41.60	37.69
	VaSE	31.04	35.63	39.03	41.39	42.57	37.93
	LazyEviction	30.56	34.44	38.40	41.11	42.57	37.42
	StacKV (ours)	**32.64**	**37.22**	**40.14**	**42.92**	**44.03**	**39.39**
GPQA-Diamond Full cache: 75.15	RPC	71.77	72.68	73.03	73.69	73.69	72.97
	R-KV	72.83	73.38	74.19	74.49	74.90	73.96
	VaSE	72.07	72.47	73.74	73.84	74.04	73.23
	LazyEviction	72.12	73.84	74.09	74.49	75.00	73.91
	StacKV (ours)	**73.28**	**74.85**	**75.05**	**75.20**	**75.40**	**74.76**
LiveCodeBench Full cache: 63.95	RPC	58.15	60.00	61.95	62.80	62.70	61.12
	R-KV	57.85	60.15	61.00	62.00	62.50	60.70
	VaSE	58.10	60.75	62.75	63.05	62.35	61.40
	LazyEviction	58.20	60.30	62.25	62.95	63.35	61.41
	StacKV (ours)	**60.15**	**62.75**	**63.20**	**63.65**	**63.95**	**62.74**

Table 9: Full results for Nemotron-9B at five retention ratios ρ.

Benchmark	Method	$1/16$	$1/8$	$1/4$	$3/8$	$1/2$	Ave.
AIME 2024 Full cache: 70.05	RPC	58.49	62.86	67.08	69.22	70.00	65.53
	R-KV	56.04	62.45	67.08	67.97	69.53	64.61
	VaSE	59.74	62.34	66.98	69.17	**70.21**	65.69
	LazyEviction	57.60	62.66	65.94	68.07	70.05	64.86
	StacKV (ours)	**61.35**	**65.00**	**68.54**	**70.00**	**70.21**	**67.02**
AIME 2025 Full cache: 57.40	RPC	43.96	49.38	52.60	54.27	56.93	51.43
	R-KV	41.61	44.64	50.05	55.73	56.61	49.73
	VaSE	42.66	49.22	53.02	55.10	57.03	51.41
	LazyEviction	38.49	46.77	51.09	53.54	57.03	49.38
	StacKV (ours)	**45.83**	**51.25**	**55.78**	**56.82**	**57.14**	**53.36**
HMMT-90 Full cache: 40.35	RPC	28.82	31.81	36.04	36.74	37.85	34.25
	R-KV	28.19	31.46	36.04	36.88	37.36	33.99
	VaSE	28.82	32.64	36.88	37.22	37.64	34.64
	LazyEviction	28.33	31.18	35.00	36.32	37.29	33.62
	StacKV (ours)	**29.86**	**34.10**	**37.99**	**38.54**	**39.17**	**35.93**
GPQA-Diamond Full cache: 60.51	RPC	54.75	56.11	58.23	58.74	59.44	57.45
	R-KV	54.29	55.76	56.97	58.48	58.99	56.90
	VaSE	55.40	56.62	58.18	59.09	59.04	57.67
	LazyEviction	54.95	57.12	58.28	59.19	59.04	57.72
	StacKV (ours)	**57.37**	**58.54**	**59.19**	**59.70**	**60.35**	**59.03**
LiveCodeBench Full cache: 82.30	RPC	68.95	74.00	78.70	79.95	81.55	76.63
	R-KV	69.55	72.95	78.75	78.50	81.00	76.15
	VaSE	70.30	74.45	79.35	79.35	80.20	76.73
	LazyEviction	68.35	74.00	76.90	79.55	80.30	75.82
	StacKV (ours)	**71.05**	**75.60**	**80.10**	**80.90**	**82.30**	**77.99**

Table 10: Full results for Kimi-Linear-48B-A3B-Instruct using the Qwen-selected parameters at five retention ratios ρ.

Benchmark	Method	1/16	1/8	1/4	3/8	1/2	Ave.
AIME 2024 Full cache: 65.83	RPC	40.00	47.29	58.49	61.77	63.39	54.19
	R-KV	38.80	45.36	58.85	61.25	63.39	53.53
	VaSE	40.89	50.00	57.81	61.61	64.53	54.97
	LazyEviction	39.90	49.27	58.49	63.18	61.72	54.51
	StacKV (ours)	**42.55**	**53.18**	**61.98**	**65.10**	**66.30**	**57.82**
AIME 2025 Full cache: 52.40	RPC	36.77	43.54	48.54	49.84	52.19	46.18
	R-KV	36.98	41.98	46.15	49.48	52.03	45.32
	VaSE	37.19	43.02	48.65	50.99	52.29	46.43
	LazyEviction	37.71	40.31	47.40	49.58	52.19	45.44
	StacKV (ours)	**39.90**	**45.52**	**50.94**	**51.67**	**52.34**	**48.07**
HMMT-90 Full cache: 41.81	RPC	25.49	29.24	34.24	36.88	38.06	32.78
	R-KV	21.18	28.33	33.54	37.57	39.10	31.94
	VaSE	25.42	31.04	34.51	35.35	36.81	32.63
	LazyEviction	25.56	29.51	35.07	35.97	36.74	32.57
	StacKV (ours)	**26.67**	**31.81**	**35.97**	**41.46**	**42.57**	**35.70**
GPQA-Diamond Full cache: 64.65	RPC	60.00	60.61	62.73	63.03	63.79	62.03
	R-KV	60.05	60.91	63.18	63.64	64.80	62.52
	VaSE	61.36	61.72	63.23	63.54	64.19	62.81
	LazyEviction	61.52	62.12	62.37	62.98	63.23	62.44
	StacKV (ours)	**62.53**	**63.23**	**64.24**	**64.85**	**65.15**	**64.00**
LiveCodeBench Full cache: 64.55	RPC	57.45	58.80	61.55	63.10	63.65	60.91
	R-KV	57.20	60.60	61.55	63.00	63.45	61.16
	VaSE	58.45	60.45	62.35	64.20	64.75	62.04
	LazyEviction	56.75	58.95	60.05	63.50	64.60	60.77
	StacKV (ours)	**60.00**	**61.60**	**64.05**	**64.50**	**66.30**	**63.29**

H Memory Measurements and Accounting

Setup. We profile Qwen3.5-4B in HF with a fixed batch size of 48, an input length of 128 tokens, and a generation length of 32,768 tokens. We use a compression interval of $P = 2048$ generated tokens and a protected recent-query window of $R = 128$ tokens.

We measure memory only in HF. Nano-vLLM preallocates its KV memory pool before inference, so compression can reduce the number of KV slots actually occupied by retained entries but cannot shrink the allocated pool or total device-memory footprint. Its whole-device memory therefore does not reveal the reduction in occupied KV slots.

Table 11: HF memory measurements: sampled whole-device peak and additional memory overhead introduced by compression.

Method	Retention ρ	Peak (GiB)	Overhead (MiB)
FullCache	1	82.33	0
RPC	1/16	10.76	30.36
RPC	1/8	16.41	36.42
RPC	1/4	24.64	43.20
RPC	3/8	33.28	49.02
RPC	1/2	45.82	57.96
StacKV	1/16	10.91	31.68
StacKV	1/8	16.47	37.86
StacKV	1/4	24.64	44.28
StacKV	3/8	33.29	50.64
StacKV	1/2	45.82	59.76

Results. Table 11 shows that the HF whole-device peak increases with retention, from 10.76–10.91 GiB at $\rho = 1/16$ to 45.82 GiB at $\rho = 1/2$, compared with 82.33 GiB for FullCache. At $\rho = 1/4$, RPC and StacKV have the same 24.64 GiB whole-device peak.

The additional memory overhead introduced by compression also increases with retention: 30.36–57.96 MiB for RPC and 31.68–59.76 MiB for StacKV. Relative to RPC, StacKV adds at most 1.80 MiB of compression overhead, while its whole-device peak differs by at most 0.15 GiB across the tested retention levels.

I Case Study: Temporal Complementarity and Spatial Protection

Figure 6 compares RPC and StacKV on a geometry problem using Qwen3.5-4B. Both displayed histories contain the relation $AI = r/\sin(A/2)$ and the intermediate result $\sin^2(A/2) = 3/13$. StacKV retains more tokens in this displayed reasoning segment than RPC: alongside the attention-selected expression, its low-magnitude selections preserve parts of the half-angle formula and the intermediate value. In the subsequent reasoning, RPC incorrectly uses $AI = r\cot(A/2)$, obtains $\sin^2(A/2) = 3/16$, and returns the incorrect product $AB \cdot AC = 504$. StacKV recalls the vertex–incenter relation, reconfirms $\sin^2(A/2) = 3/13$, and reaches the correct answer, 468.

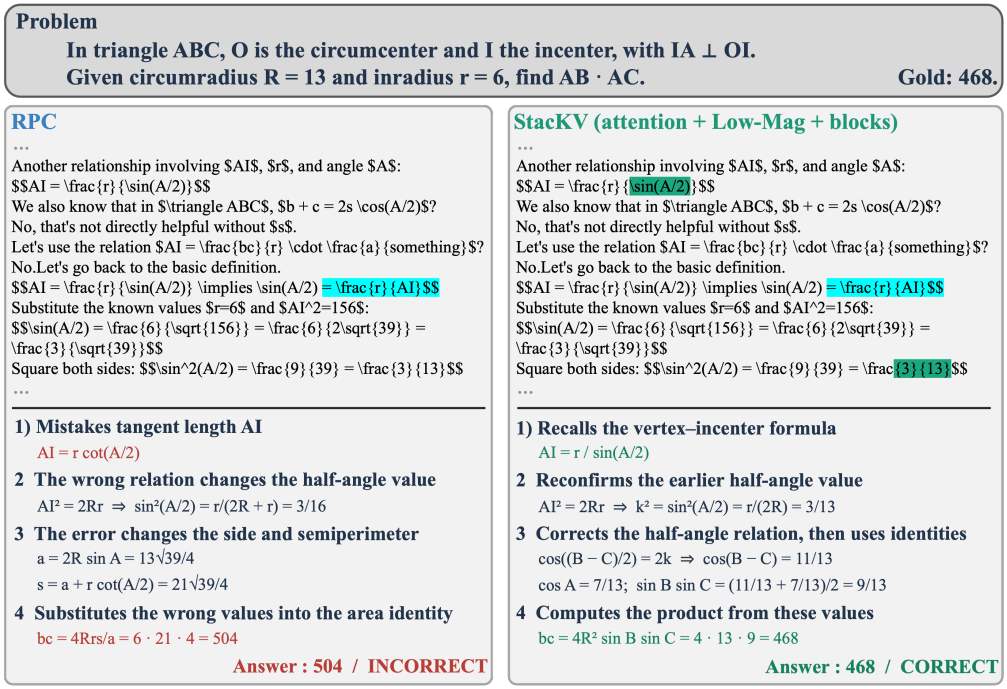

Figure 6: Case study of KV-cache compression on a geometry problem with Qwen3.5-4B. Token highlights summarize compression selections across layers: blue denotes tokens selected by attention scores, and green denotes low-magnitude (Low-Mag) tokens selected by the recorded linear-update signal. StacKV retains more tokens in the displayed segment, illustrating local protection through block-wise budgets together with complementary Low-Mag selection. The lower panels summarize the subsequent reasoning: RPC returns 504, whereas StacKV returns the correct answer, 468.

Spatial protection. The additional retained tokens in this segment illustrate the role of block-wise allocation. Global attention ranking can allocate little capacity to an earlier reasoning segment when other regions receive higher scores. StacKV reserves length-proportional quotas for contiguous blocks and selects tokens within each block, protecting local capacity under the overall retention budget. Finer blocks toward older history further constrain gaps in coverage after repeated com-

pression. Here, this protection leaves more of the displayed derivation available for subsequent reasoning.

Temporal complementarity. Within these local budgets, the linear signal helps determine which tokens survive. Recent-query attention reflects the current observation window, while recorded linear updates supply a historical view. The green highlights show how Low-Mag selection supplements attention with earlier mathematical details that are reused later. Together with the controlled comparisons in Table 1, this case illustrates how spatial protection and temporal complementarity work together: blocks preserve capacity across the reasoning history, and the linear signal provides a key temporal complement to attention within that capacity.