GATESIGHT: RECURRENT MEMORY ANTICIPATES ATTENTION MEMORY FOR QUERY-AGNOSTIC KV EVICTION IN HYBRID LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Hybrid language models interleave Gated DeltaNet (GDN) layers, whose recurrent memory has a fixed size, with full-attention layers, whose key–value (KV) cache grows with the context and dominates serving memory. In this paper, we ask whether the recurrent memory already knows what the attention memory will need. We first show that which GDN heads of Qwen3.8-27B keep long memories is fixed by the weights rather than by the document. Next, we define the future utility of a KV entry by how much removing it changes the attention output over the model's own continuations, and find that the write gate, decay and write magnitude that GDN layers emit during prefill anticipate this utility before any query is seen. Because the relation is spread across many GDN heads and layers, and a linear map recovers its trend but less of its high-utility tail than a nonlinear one, we propose GATESIGHT, which learns it with lightweight per-head multilayer perceptrons that read these signals and no attention score. Their predicted utility decides which entries each attention layer keeps, and how much each layer keeps is set by the model's native attention output gate, whose mean differs by more than an order of magnitude across layers. Finally, at serving time, GATESIGHT scores the cache once after prefill and physically releases discarded KV entries. Experimental results on SCBench and RULER-4096 with Qwen3.8-27B and, with the same recipe, the sparse-attention model Qwen3.8-Flash-Next, up to the 262K-token limit of the models, show that with a tenth of the attention KV cache, GATESIGHT is the most accurate of five query-agnostic methods on three of the four model–benchmark pairs and leads KVzip on all four, by 9.5 to 47.2 points. Notably, it scores a full-length context with negligible overhead relative to prefill.

1 INTRODUCTION

When a Transformer generates text, each attention layer caches the keys and values of all previous tokens so that they are not recomputed at every step, and this key–value (KV) cache grows linearly with the context. Recently, hybrid models have curbed this growth: Qwen3.8, Kimi Linear and Jet-Nemotron replace most of the attention layers with Gated DeltaNet (GDN) layers or a variant of them, which compress the whole context into a recurrent state whose size is independent of the context length (Yang et al., 2025; Qiu et al., 2026; Kimi Team et al., 2025; Gu et al., 2025). These hybrids, however, keep a minority of attention layers for retrieval (Qiu et al., 2026; Kimi Team et al., 2025; Gu et al., 2025). These layers still store a token-indexed cache, the only per-sequence state that grows with the context. In Qwen3.8-27B, for example, this cache exceeds the GDN states beyond about 2.4K tokens and is about 42 times larger at 100K tokens (BF16 cache, FP32 recurrent states), which limits the number of sessions that a GPU can serve concurrently (Kwon et al., 2023). A natural remedy is therefore to keep only the KV entries that will be needed later and to evict the rest.

However, it is challenging to determine in advance which entries will be needed. Methods that score entries by observed attention require a query (Zhang et al., 2023; Li et al., 2024). We focus on contexts that are prefilled once and whose cache is then reused to answer every subsequent question, a reuse pattern that inference frameworks and model providers have adopted (Li et al., 2025). In this setting, the cache occupies GPU memory from the end of prefill and must therefore be compacted at that point, and a single compacted cache serves questions that have not yet been asked (Kim et al.,

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.

2025). Consequently, the eviction scores cannot depend on these questions. Existing query-agnostic methods follow one of two routes. The first runs the model a second time: KVzip reconstructs the context from its own cache and keeps the entries that the reconstruction attends to (Kim et al., 2025), an extra pass whose attention cost grows quadratically with the context length under dense attention. The second, adopted by KeyDiff and TriAttention, scores the cached keys offline. KeyDiff keeps the keys that are least similar to the mean key of their head (Park et al., 2025). TriAttention derives a preference over relative distances from the per-head centres of queries and keys, defined only on the dimensions that RoPE rotates, and scores each key by its position under this preference and by its norm (Mao et al., 2026). Both routes read only the attention layer's own cache.

In this paper, we instead use the recurrent memory of a hybrid model to select which entries of its attention memory to keep. The model maintains this memory through its GDN layers, which decide, token by token, how strongly to write each token into the memory and how quickly to forget it. This raises a natural question: do these decisions carry information about which exact KV entries the attention layers will later need? In other words, does the recurrent memory already know what the attention memory will need? To answer this question, we instrument Qwen3.8-27B and highlight three empirical observations, which in turn motivate our method.

Observation 1. GDN memory dynamics are structured and stable across documents. Heads differ by orders of magnitude in how long they retain information, and which heads are slow is a property of the weights that is stable across documents (§3.1).

Observation 2. GDN write and forget signals anticipate future KV utility. The write gate, the decay and the write magnitude that GDN layers emit during prefill predict the future utility of KV entries before any query exists, on short and long documents and on retrieval tasks alike (§3.2).

Observation 3. The relation is distributed and nonlinear. No single signal, head or layer captures it reliably, and a linear map recovers its trend but less of its high-utility tail than a nonlinear one (§3.3).

Based on these observations, we propose GATESIGHT, a compact estimator of future KV utility that reads the signals the GDN layers already compute. Specifically, for every attention layer and KV head, a small multilayer perceptron (MLP), trained once on generic text, maps these signals and the age of an entry to a utility score after prefill, without materialising attention scores. The estimator decides which entries each attention layer keeps, and the model's native attention output gate sets how many, in proportion to its mean value, which differs by more than an order of magnitude across layers. The pruned cache is physically compacted and can be reused across the questions that follow.

We implement GATESIGHT in vLLM and evaluate it on SCBench and RULER-4096 with Qwen3.8-27B and the sparse-attention model Qwen3.8-Flash-Next, up to the 262K-token limit of the models. Experimental results show that with a tenth of the attention KV cache, GATESIGHT is the most accurate of five query-agnostic methods on three of the four model–benchmark pairs and leads KVzip on all four, by 9.5 to 47.2 points. Notably, it scores a full-length context of Qwen3.8-27B in 0.246 s, about 0.5% of the prefill time.
Our key contributions can be summarized as follows:

- We present three observations on the GDN memory dynamics of a production hybrid model (§3).
- We propose GATESIGHT, a query-agnostic KV-eviction method for hybrid models that predicts future KV utility from native GDN signals, without attention scores, and sets the budget of each layer from the attention output gate.
- We evaluate GATESIGHT against four query-agnostic methods on two hybrid models, and show in an ablation on Qwen3.8-27B that, averaged over its tasks, the gate-based allocation is at least as accurate as flat and PyramidKV ones at every kept fraction.

2 BACKGROUND AND PROBLEM SETTING

Hybrid GDN–attention models. Qwen3.8-27B, our main model, has 64 layers: 48 are GDN layers and 16, every fourth layer, are full-attention layers (Qwen Team, 2026b). These attention layers apply a sigmoid output gate before the output projection.[1]

[1] Implementation: `Qwen3NextAttention.forward`, vLLM v0.28.0.

Native GDN signals. A GDN layer keeps, for each of its 48 value heads (three per key head), a matrix state $S_t \in \mathbb{R}^{128\times128}$ updated with the gated delta rule (Yang et al., 2025; 2024; Schlag et al., 2021),

$$S_t = \alpha_t\big(I - \beta_t k_t k_t^\top\big)S_{t-1} + \beta_t k_t v_t^\top, \qquad \alpha_t = \exp(g_t),\ g_t \le 0, \tag{1}$$

where k_t and v_t are the key and value written by token t, $\beta_t \in (0,1)$ is the write gate and g_t the log-decay. For every token and every GDN head, the forward pass produces the write gate β_t and the decay g_t directly, and the value vector v_t, whose norm $\|v_t\|$ we call the write magnitude. We call these three scalars the native GDN signals; obtaining them needs no additional forward pass, only the norm of a vector the prefill has already produced.

Attention KV cache. A full-attention layer ℓ of Qwen3.8-27B has 24 query heads and 4 KV heads (grouped-query attention, GQA (Ainslie et al., 2023), six query heads per KV head) with a head dimension of 256 (Qwen Team, 2026b), and stores, for each KV head h, the cache $\mathcal{C}_{\ell,h} = \{(k_{\ell,h,i}, v_{\ell,h,i})\}_{i=1}^{n}$ of a context of n tokens. Every token adds one entry per layer and head, so the cache grows linearly with n: $16 \times 4 \times 2 \times 256$ values per token, 64 KiB in bfloat16. The entry is the unit of eviction: removing $(k_{\ell,h,i}, v_{\ell,h,i})$ removes position i from the softmax of all six query heads of the group, and the GDN states are left untouched.

Query-agnostic KV eviction. Formally, given a kept fraction p and a protected instruction prefix of prot tokens (at most 256, kept by every method), the model-wide budget, exclusive of the protected prefix, is $B = 16 \cdot 4 \cdot \mathrm{round}\big(p\,(n - \mathrm{prot})\big)$ entries, and a method chooses kept sets $S_{\ell,h} \subseteq \{\mathrm{prot}+1, \dots, n\}$ with $\sum_{\ell,h} |S_{\ell,h}| \le B$. The choice splits into two decisions: which entries a layer keeps (within-layer selection) and how many entries each layer receives (cross-layer allocation).

Future KV utility. To study what an entry is worth before a query exists, we use the model's own continuations as a stand-in for its possible futures. The resulting utility is the analysis and training target of this paper; it is never computed at serving time, where the estimator of §4 predicts it from the native signals. For a context of n tokens, we sample $M = 4$ continuations $z^{(1)}, \dots, z^{(M)}$ of up to 128 tokens from the model with its default sampling parameters (Appendix B). For a full-attention layer, a KV head h and a context entry i, the influence of i on a continuation token t is the change in the output of that layer after the output projection, with the output gate left out, when entry i is removed from the softmax of every query head j of the GQA group $\mathcal{Q}(h)$,

$$\Delta r(t,i) = \sum_{j\in\mathcal{Q}(h)} \frac{a_j(t,i)}{1 - a_j(t,i)}\, W_O^{(j)}\big(v_i - o_j(t)\big), \qquad U(i) = \max_{t\in z^{(1..M)}} \big\|\Delta r(t,i)\big\|, \tag{2}$$

where $a_j(t,i)$ is the attention weight of query t on entry i, $o_j(t)$ the output of head j and $W_O^{(j)}$ its slice of the output projection. With the queries, the other entries and the rest of the forward pass held fixed, the expression is the exact joint leave-one-out change of the ungated output of the layer (Appendix B). The gate is left out because it changes the ranking of a head's entries by almost nothing (Appendix D), and without it the closed form is evaluated once per entry (Appendix B). The objects are now in place: the native GDN signals of a context on one side, the KV entries of its attention layers on the other. The next section asks whether the former anticipate the utility of the latter.

3 Three observations about recurrent memory in a hybrid model

In this section, we examine whether the recurrent memory of a hybrid model anticipates what its attention memory will need. To this end, we instrument Qwen3.8-27B on 2,428 documents of 750 to 1,250 tokens drawn from nine subsets of the Nemotron pretraining sample (NVIDIA, 2025b) (Appendix C). For every document, we record the native GDN signals of all 48 layers and 48 heads during one prefill and compute the utility $U(i)$ of every KV entry of every full-attention layer and KV head with Eq. 2.

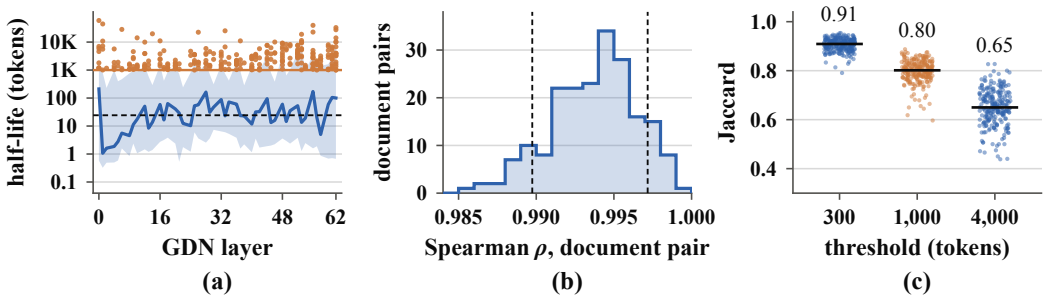

Figure 1: GDN memory dynamics of Qwen3.8-27B are structured and stable across documents. (a) Median over heads (line), 10th–90th percentile (band), global median (dashed), slow heads and threshold (orange). (b) Half-life ordering of heads, compared per document pair; dashed: 10th and 90th percentiles. (c) Per-document against global slow set, one dot per document; bars: medians.

3.1 Observation 1: GDN memory dynamics are structured and stable across documents

GDN heads span multiple orders of magnitude in their memory time scales, and these differences remain highly stable across documents. To quantify the time scale of a head, we compute its memory half-life $\ln 0.5/\bar{g}$, where $\bar{g}$ is the mean log-decay over a document, and Figure 1a plots its median over documents. Across heads the median is about 24 tokens, but every layer spans heads that forget within a handful of tokens and heads that hold information for hundreds to thousands of tokens. Moreover, which heads are slow is nearly fixed. Between random pairs of the 240 analysis documents, the ordering of the 2,304 heads by half-life agrees at Spearman 0.99 to 0.997 (Figure 1b). The weights alone predict this ordering: the static half-life read from the checkpoint matches the measured one at Spearman 0.942 (Appendix C).

The text moves the boundary of the slow set rather than its core. As shown in Figure 1c, at a threshold of 1,000 tokens the per-document slow set overlaps the global slow set at Jaccard 0.80. This is why the slow-head set is fixed once and reused for every context (§3.2).

Beyond the decay, the write side is likewise dominated by a static head identity, which accounts for 71.7% of the variance of the log effective write $\beta\|v\|$, against 10.2% for the token (Appendix C). These stable roles motivate the estimator of §4.1, which attaches a fixed weight to every GDN head.

3.2 Observation 2: GDN write and forget signals anticipate future KV utility

Observation 1 identifies stable head roles with token-dependent variation on top of them. We now ask whether this variation tracks what the attention layers will need. To this end, for every document, full-attention layer and KV head, we rank-correlate $U(i)$ across entries with summaries of the native write gate β, forgetting $-g$, write magnitude $\|v\|$ and effective write $\beta\|v\|$ over the up to six preceding GDN layers (§4.1), and with the two write magnitudes over slow heads alone ($\|v\|_{\mathrm{s}}$, $\beta\|v\|_{\mathrm{s}}$). Following the half-life criterion of Observation 1, we fix the 293 heads whose median half-life over all 2,428 documents is at least 1,000 tokens; the set is never re-estimated per document. As shown in Figure 2a, every summary is positively correlated with U in each of the nine text subsets. The 16-layer mean correlations are informative rather than strong, ranging from $+0.09$ to $+0.23$. Notably, restricting the same effective write to slow heads more than doubles its correlation, from $+0.10$ to $+0.23$. This indicates that the stable head structure helps locate predictive information, which motivates reading heads with separate weights rather than only through a window mean.

Along depth, the slow-head $\beta\|v\|$ is near zero in the first two attention layers and reaches $+0.19$ to $+0.33$ from the third on (Figure 2b). Entries in its highest decile reach the top utility decile with probability 18.2% (1.82× the 10% chance rate), and entries in its lowest decile reach the bottom utility decile with probability 19.2% (1.92× chance; Figure 2c). In other words, the entries that the slow heads, whose memory half-life exceeds a thousand tokens, write strongly tend to be the ones the attention layers will need. Conversely, the entries these heads leave almost unwritten tend to be the ones attention will not need. On the forget side, forgetting $-g$ is positively correlated with U in every

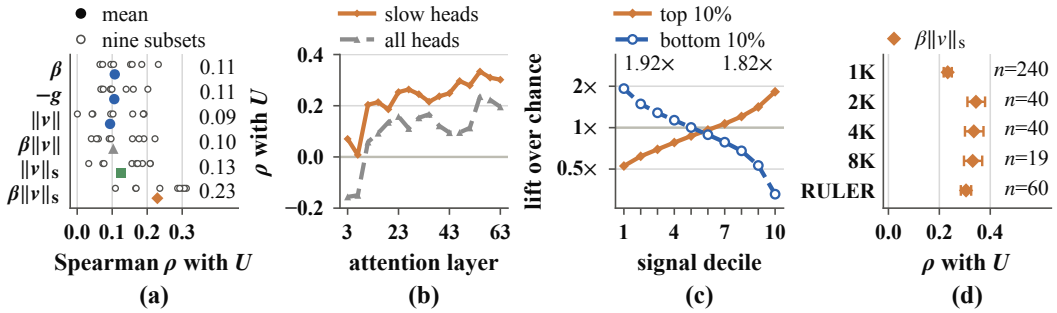

Figure 2: Native GDN signals anticipate future KV utility U (Qwen3.8-27B). (a) Unqualified signals use the window mean; subscript s: slow heads. (b) Effective write, slow versus all heads. (c) $\beta\|v\|_s$: top/bottom utility-decile probabilities over 10% chance. (a, c) average 16 layers; (a–c) average four KV heads. (d) n: documents; bars: 95% bootstrap CIs over documents.

attention layer (+0.06 to +0.19). These associations suggest that useful entries tend to arrive where the recurrent state updates more strongly.

The fixed slow set also carries over to 2K to 8K documents and to RULER-8K contexts (Hsieh et al., 2024) without re-estimation (Figure 2d; Appendix C).

3.3 Observation 3: The relation is distributed and nonlinear

A natural starting point is to rank KV entries by a single native GDN signal. However, Table 1 shows the limits of this approach and what the two learned predictors add. Ridge regression and the per-head MLP read the same 867 window features (§4.1) and are both evaluated leave-one-corpus-out.

Many heads carry predictive signal, but no single head provides a strong ranking. Inside the window of GDN layers preceding an attention layer, 40 to 70% of the 288 (layer, head) cells correlate with U at $|\rho| > 0.1$, yet no cell exceeds 0.39, and which GDN layers matter changes with depth (Figures 6 and 7, Appendix C). Consequently, of the true top-10% entries, the best fixed heuristic (age, older first) recovers 19.2%, the slow-head $\beta\|v\|$ 18.2%, and the single GDN head selected with the true utility for each attention layer and KV head, an in-sample ceiling for one-cell rules, 22.6%.

Table 1: Predicting U from native GDN signals (mean over layers and heads). ρ: Spearman; Ov@10: top-10% overlap (%). Best in bold.

Predictor	ρ (↑)	Ov@10 (↑)
Untrained signals		
random	+0.000	10.0
age	+0.071	19.2
slow-head $\beta\|v\|$	+0.229	18.2
single-head oracle	+0.232	22.6
Learned, 867 features		
ridge	**+0.548**	38.6
MLP (GateSight)	+0.543	**41.0**

Finally, the nonlinear model improves recovery of high-utility entries despite a slightly lower overall rank correlation. Ridge regression reaches a Spearman of 0.548 against 0.543 for the per-head MLP. However, the MLP recovers 41.0% of the true top-10% entries against 38.6% for ridge and covers 58.4% against 55.8% of their utility mass, and it is ahead in all sixteen attention layers on both measures (Table 5; Figure 8). Under aggressive budgets it is exactly this tail that decides which entries survive.

4 GateSight: A learned cross-memory utility estimator

We now turn the observations of §3 into an eviction method, GateSight (Figure 3). GateSight consists of three parts: (i) an estimator, which decides what to keep within a layer and is the part the observations determine; (ii) an allocation, which decides how much each layer keeps from a signal of the attention layer itself, its output gate, since the observations leave this question open; and (iii) a deployment, which compacts the cache and serves it to later questions.

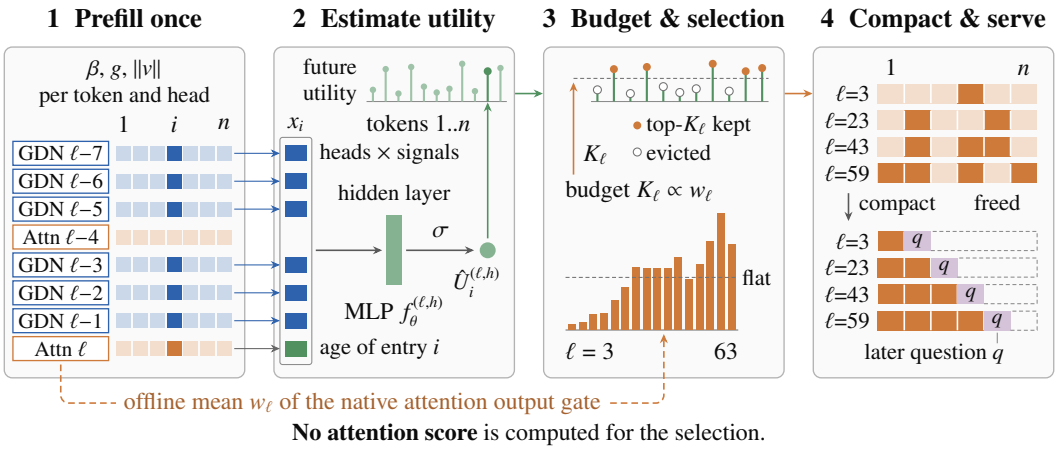

Figure 3: GATESIGHT for attention layer ℓ and KV head h. (1) One prefill of the context. (2) A per-head MLP, trained offline, predicts the future utility of each entry. (3) Each head keeps its top-K_ℓ entries under the gate-proportional layer budget. (4) Kept entries are packed into the front pages, and the freed tail returns to the block pool.

4.1 ESTIMATOR

The estimator maps the native GDN signals and the age of a context entry to its predicted utility. Specifically, for a full-attention layer ℓ and a KV head h, the input for context entry i is

$$x_i^{(\ell)} = \Big[\{\beta_{i,m}^{(j)},\ g_{i,m}^{(j)},\ \|v_{i,m}^{(j)}\|\}_{j\in W(\ell),\, m=1..48}\ ;\ \tfrac{i}{n},\ \tfrac{n-i}{n},\ \tfrac{\log(1+n-i)}{\log(1+n)}\Big], \tag{3}$$

the native signals of all 48 heads of the up to six GDN layers $W(\ell)$ that precede ℓ (three for the first attention layer), plus three normalised age features: $6 \times 48 \times 3 + 3 = 867$ dimensions. The window is local although the best single GDN layer is not always inside it (Figure 7): for attention layers 39 to 51 that layer is GDN layer 28, and the best layer inside the window correlates with U less than 0.07 lower; a fixed window gives every estimator the same input at every depth (Appendix F varies the window from 3 to 12 layers). The estimator $f_\theta^{(\ell,h)} : \mathbb{R}^{867} \to [0,1]$ is a multilayer perceptron with one hidden layer of 64 units and a sigmoid output; Qwen3.8-27B has $16 \times 4 = 64$ of them, 3.4M parameters in total. Unlike a designed importance heuristic, the estimator is a learned reader of the relation of §3.2.

Training target. We train the estimator against the rollout utility of Eq. 2, max-normalised per (document, layer, KV head) to $[0,1]$, with a binary cross-entropy loss. The training set consists of the 2,428 Nemotron documents of §3 with 500 random positions per document, and we use AdamW with a learning rate of 2×10^{-3} for 12 epochs. Training uses no task data, no questions and no downstream benchmark. Although the training corpus is generic text of about one thousand tokens, the estimator is applied unchanged to contexts of up to 262K tokens. On unseen documents of 2K to 8K tokens, it keeps a Spearman of 0.56 to 0.60 with U, and on RULER-8K contexts 0.43, recovering 41 to 45% and 41% of the top-10% entries, respectively (Table 6).

4.2 PER-LAYER ALLOCATION FROM THE OUTPUT GATE

The observations fix what to keep within a layer but not how much each layer keeps. To set the latter, we use the model's native attention output gate. Across layers the mean gate spans more than an order of magnitude, 0.022 to 0.459, while within a head it barely reorders tokens (Appendix D), so the gate is used here and only here. The layer weight w_ℓ is its mean magnitude, measured once offline: for each KV head the gate is the maximum over its six query heads, and w_ℓ is the mean over the heads of the layer. Layer ℓ then receives $K_\ell \propto w_\ell$ entries per head, with the same model-wide total as a flat allocation (Appendix G); the rule has no exponent or temperature (Appendix D).

We set the budget per layer rather than per head, because a per-head split of the same budget is no more accurate (Appendix D) and a per-layer budget is the length of a block table, which keeps the kept cache of every layer a dense prefix (§4.3). Section 5.6 compares the gate-proportional rule at the

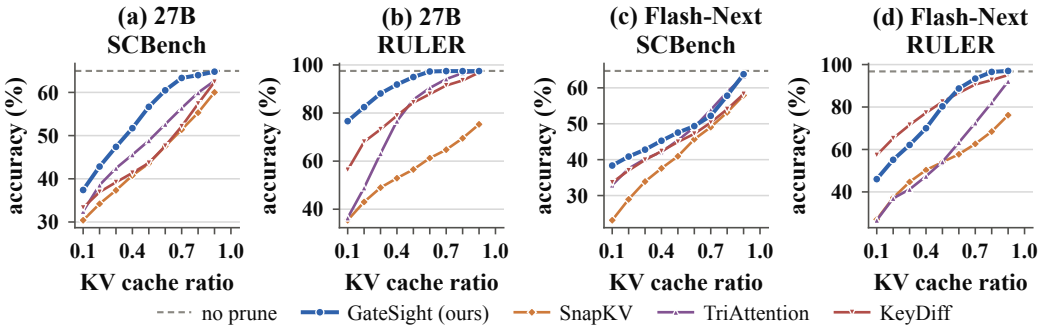

Figure 4: Task-averaged accuracy under KV pruning: (a, b) Qwen3.8-27B and (c, d) Qwen3.8-Flash-Next, each on SCBench and RULER.

same total budget with a flat allocation and with the depth-decreasing schedule of PyramidKV (Cai et al., 2025b).

4.3 Deployment in vLLM with physical compaction and session reuse

We implement GateSight inside vLLM 0.28 (Kwon et al., 2023) as a plugin that changes no kernel. Each attention layer has its own block table, so a per-layer budget is the length of that table. When prefill ends, the estimator scores the context once, each layer moves the entries it keeps to the front of its pages, and the freed blocks return to the pool, which physically releases their memory. Later questions reuse the compacted cache with the GDN states restored (Appendix G).

5 Experiments

5.1 Setup

Models. We evaluate GateSight on two hybrid models. The main model is Qwen3.8-27B (§2). To test GateSight on a sparse-attention cache, we also evaluate Qwen3.8-Flash-Next (Qiu et al., 2026), a mixture-of-experts model with 125B parameters, 6B of them active per token, whose attention layers use Qwen Sparse Attention (QSA): each query attends only to the context tokens that an indexer selects. We use its official FP8 checkpoint and retrain the estimator with the same recipe, computing the utility of Eq. 2 under this sparse attention (Appendix H).

Benchmarks and metrics. In the typical use of KV compression, a long context is compressed once, and every later turn of the conversation reads the compressed cache. SCBench (Li et al., 2025) evaluates this scenario: each context is prefilled once and followed by several questions. We run all twelve SCBench tasks in its shared-context multi-query mode. RULER (Hsieh et al., 2024) complements it with thirteen synthetic tasks of retrieval, multi-hop tracing, aggregation and question answering, at a sequence length of 4,096 tokens. All tasks use their official metrics, except that two SCBench tasks use a revised scorer (Appendix A). The kept fraction takes nine values, $p \in \{0.9, 0.8, \ldots, 0.1\}$.

Baselines and protocol. We compare with query-agnostic methods that score the prefix once after prefill, and group them by whether scoring runs another forward pass over the context. The main comparison (§5.2) covers the three methods that do not: SnapKV (Li et al., 2024), in its query-agnostic form, probes the cache with the attention of the last 32 context tokens, and TriAttention (Mao et al., 2026) and KeyDiff (Park et al., 2025) score the cached keys directly. KVzip (Kim et al., 2025) runs a second forward pass that reconstructs the context, and §5.5 compares with it together with the cost of that pass. Each uses the budget allocation its paper defines (KVzip's global score-sorted allocation, flat per-head budgets for the others); the gate-based per-layer allocation belongs to GateSight only. To ensure a fair comparison, all methods run in the same vLLM pipeline under the prefix-only protocol of Appendix A, with every setting listed in Appendix G.

5.2 MAIN RESULTS

Qwen3.8-27B. As shown in Figure 4a,b, GATESIGHT is more accurate than SnapKV, TriAttention and KeyDiff at every kept fraction on both benchmarks.

Qwen3.8-Flash-Next. Figure 4c,d report the results on Qwen3.8-Flash-Next (Appendix H). On SCBench, GATESIGHT is more accurate than SnapKV and KeyDiff at every kept fraction, while TriAttention stays within 2.2 points of it at $p \geq 0.5$ and trails it by 2.6 to 5.4 points at $p \leq 0.4$. On RULER-4096, it is more accurate than SnapKV and TriAttention at every kept fraction, whereas KeyDiff leads at $p \leq 0.5$, by up to 11.3 points. This lead comes from four needle tasks (Appendix E).

5.3 SCORING COST

We further measure the wall-clock cost of scoring in the same vLLM pipeline, for contexts of 4K to 256K tokens (Table 2). GATESIGHT gathers the GDN signals and runs 64 small networks, which is linear in n and independent of the attention computation. Like SnapKV, TriAttention and KeyDiff, it requires no additional forward pass over the context. Its scoring takes 0.5% to 2.6% of the prefill time, 0.246 s against 46.4 s at 256K tokens, and 1.0% to 1.4% on Qwen3.8-Flash-Next (Appendix H). In contrast, the second pass of KVzip takes 1.5 to 2.2 times the prefill time, 100.3 s at 256K tokens.

Table 2: Wall-clock scoring time (s) per context in the same vLLM pipeline, Qwen3.8-27B, by context length; median of three runs after a warm-up.

Method	Context length (tokens)			
	4K	16K	64K	256K
GATESIGHT	0.010	0.023	0.069	0.246
TriAttention	0.014	0.016	0.034	0.109
KeyDiff	0.004	0.007	0.023	0.084
SnapKV	0.028	0.063	0.131	0.452
KVzip	0.747	2.787	13.540	100.279

5.4 PRUNING A SPARSE-ATTENTION CACHE

In Qwen3.8-Flash-Next each query token attends to only 2,048 context tokens, but the QSA indexer that selects them scores one compressed key for every four tokens of the whole context (Qiu et al., 2026). The cache and this scan therefore grow linearly with the context while the attention does not. GATESIGHT shrinks the index itself: the kept tokens are regrouped into QSA blocks (Appendix H), so at kept fraction p the index of a layer holds $pn/4$ compressed keys on average instead of $n/4$.

5.5 COMPARISON WITH KVZIP

Table 3 compares GATESIGHT with KVzip on both models at three kept fractions and on the mean over all nine. On Qwen3.8-27B, neither method leads at every kept fraction. GATESIGHT stays within 1.2 points of KVzip at $p \geq 0.7$ without the second pass KVzip runs over the context (Table 2). KVzip leads at intermediate fractions, by up to 8.7 points on SCBench and 3.6 on RULER-4096 at $p = 0.4$, but the order reverses at $p = 0.1$, where GATESIGHT leads by 9.5 points on SCBench and 47.2 on RULER-4096. Per task, GATESIGHT matches or exceeds KVzip on at least 15 of the 25 tasks at every kept fraction, and on 22 at $p = 0.1$.

Table 3: Comparison with KVzip: task-averaged accuracy (%); Mean: nine kept fractions. Higher of the two methods in bold.

Benchmark	Method	$p = 0.7$	0.4	0.1	Mean
Qwen3.8-27B					
SCBench	GATESIGHT	63.3	51.7	**37.4**	54.3
	KVzip	**64.5**	**60.4**	27.9	**56.2**
RULER	GATESIGHT	97.4	91.9	**76.6**	**91.5**
	KVzip	**97.7**	**95.5**	29.4	83.4
Qwen3.8-Flash-Next					
SCBench	GATESIGHT	52.2	45.3	**38.4**	48.7
	KVzip	**64.4**	**60.9**	23.0	**54.2**
RULER	GATESIGHT	93.4	70.0	**46.0**	**76.6**
	KVzip	**96.7**	**79.6**	20.5	74.0

On Qwen3.8-Flash-Next, KVzip leads from $p = 0.3$ on SCBench and from $p = 0.4$ to 0.8 on RULER-4096, by up to 15.7 and 11.0 points at $p = 0.5$, and GATESIGHT leads at $p = 0.1$, by 15.4 and 25.6 points. Per task, GATESIGHT matches or exceeds KVzip on at least 14 of the 25 tasks at every kept fraction, and on 23 at $p = 0.1$. On both models, KVzip's lead on SCBench comes from the tasks whose answer is copied verbatim from the context (Appendix E).

5.6 Ablations

To isolate the contribution of the per-layer budget rule of §4.2, we conduct a controlled comparison at a fixed total budget. Specifically, on three short-answer SCBench tasks (`kv`, `qa_eng`, `many_shot`) and three RULER-4096 tasks (Figure 13), with every context and all nine kept fractions, each context is prefilled once and scored once by the estimator, and three compacted children are forked from the same scores: the gate-proportional rule of GateSight, a flat allocation with the same budget in every layer, and the PyramidKV schedule (Cai et al., 2025b), which decreases the budget arithmetically with depth ($\beta = 20$; Appendix F).

Table 4: Allocation ablation on Qwen3.8-27B: accuracy (%) over three SCBench and three RULER tasks; Mean: nine kept fractions. Best in bold.

Allocation	$p = 0.7$	0.4	0.1	Mean
SCBench				
gate-proportional	**72.4**	**57.6**	**39.2**	**60.6**
flat	61.2	48.4	38.6	53.7
PyramidKV	43.8	38.7	37.8	44.3
RULER				
gate-proportional	**100.0**	**100.0**	**98.6**	**99.8**
flat	**100.0**	**100.0**	95.5	99.4
PyramidKV	99.9	97.2	52.1	90.4

The three children keep exactly the same number of entries, so any difference comes from where the entries sit. Table 4 reports the accuracy at three kept fractions and the mean over all nine, and Figure 13 gives each task. Averaged over the tasks of each benchmark, the gate-proportional rule is at least as accurate as the other two at every kept fraction. On SCBench, its lead comes from `kv`, by 20.7 points over flat and 47.8 over PyramidKV on the nine-budget mean, while the three rules differ by about one point or less on `qa_eng` and `many_shot`. On RULER-4096, at $p = 0.1$ the gate-proportional rule reaches 98.6%, against 95.5% for flat and 52.1% for PyramidKV.

6 Related work

One line of work scores entries by the attention they have received, from heavy hitters (Zhang et al., 2023), persistence across decoding steps (Liu et al., 2023) and observation windows (Li et al., 2024) to per-head policies (Ge et al., 2024), redundancy terms (Cai et al., 2025a) and budgets that vary across layers and heads (Cai et al., 2025b; Feng et al., 2025; Fu et al., 2025). However, these methods need queries or a probe over the context. StreamingLLM preserves attention sinks and a recent window (Xiao et al., 2024), whereas DuoAttention exploits a static partition into retrieval and streaming heads (Xiao et al., 2025); Observation 1 finds an analogous static structure in the recurrent layers. Other query-agnostic methods either reconstruct the context in a second pass, as KVzip does (Kim et al., 2025), or score the cached keys directly, as TriAttention, KeyDiff and the L_2 norm of keys do (Mao et al., 2026; Park et al., 2025; Devoto et al., 2024). Compared with these methods, GateSight is query-agnostic like KVzip and probe-free like TriAttention and KeyDiff, but differs from all three in reading a second memory system of the model, the recurrent state, instead of the attention layer itself. Hybrid models interleave recurrent layers such as Mamba and Gated DeltaNet with attention (Gu & Dao, 2024; Dao & Gu, 2024; Yang et al., 2025; Lieber et al., 2024; Ren et al., 2025; NVIDIA, 2025a; De et al., 2024; MiniMax et al., 2025; Qwen Team, 2026a; Qiu et al., 2026); to the best of our knowledge, no prior work uses these recurrent layers to manage the KV cache of the attention layers (Appendix I).

7 Conclusion

In this work, we show that the recurrent memory of a hybrid language model knows, to a useful degree, what its attention memory will need: the write and forget decisions of Gated DeltaNet layers are structured, stable across texts, and predictive of which KV entries will later change the attention output. Because the relation is distributed and nonlinear, we present GateSight, which reads it with a compact learned estimator and evicts KV entries once after prefill, without a query or an attention score. The physically compacted cache can then be reused across later questions. Experimental results on SCBench and RULER-4096 with Qwen3.8-27B and Qwen3.8-Flash-Next show that with a tenth of the attention KV cache, GateSight is the most accurate of five query-agnostic methods on three of the four model–benchmark pairs and leads KVzip on all four, by 9.5 to 47.2 points.

AI USE STATEMENT

We used large language model assistants for two tasks: drafting and editing the text of the paper, including rewording sentences and drafting result paragraphs from the measured numbers; and writing and debugging code, including the evaluation harness, the vLLM plugins, the training and labelling scripts, and the scripts that generate the figures and tables. Generative AI was not used to generate research ideas or to design the method or the experiments. Every reported number is produced by scripts from the raw result files and was checked against them; the authors reviewed all AI-assisted text, tested AI-written code before use, and take full responsibility for the content of the paper.

REPRODUCIBILITY STATEMENT

The complete code for training and evaluating GATESIGHT is released in an anonymous repository (https://anonymous.4open.science/r/gatesight-anonymous-5226/): the vLLM plugins for both models, the training and labelling scripts, the evaluation scripts for SCBench and RULER-4096, the trained estimators with their calibration artifacts, and the pinned software environment. Section 5.1 and Appendix G list every hyperparameter, budget rule and engine setting; Appendix A states the exact prefix-only evaluation protocol shared by all methods; Appendix C gives the definitions behind the three observations.

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 *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4895–4901, 2023. arXiv:2305.13245.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, pp. 60980–61005, 2025a. arXiv:2505.24133.

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. In *Conference on Language Modeling (COLM)*, 2025b. arXiv:2406.02069.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *International Conference on Machine Learning (ICML)*, volume 235 of *PMLR*, pp. 10041–10071, 2024. arXiv:2405.21060.

Soham De, Samuel L. Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, et al. Griffin: Mixing gated linear recurrences with local attention for efficient language models. *arXiv preprint arXiv:2402.19427*, 2024.

Alessio Devoto, Yu Zhao, Simone Scardapane, and Pasquale Minervini. A simple and effective L_2 norm-based strategy for KV cache compression. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 18476–18499, 2024. arXiv:2406.11430.

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)*, volume 38, pp. 113152–113188, 2025. arXiv:2407.11550.

Yu Fu, Zefan Cai, Abedelkadir Asi, Wayne Xiong, Yue Dong, and Wen Xiao. Not all heads matter: A head-level KV cache compression method with integrated retrieval and reasoning. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2410.19258.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive KV cache compression for LLMs. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.01801.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *Conference on Language Modeling (COLM)*, 2024. arXiv:2312.00752.

Yuxian Gu, Qinghao Hu, Haocheng Xi, Junyu Chen, Shang Yang, Song Han, and Han Cai. Jet-Nemotron: Efficient language model with post neural architecture search. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, pp. 47191–47218, 2025. arXiv:2508.15884.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *Conference on Language Modeling (COLM)*, 2024. arXiv:2404.06654.

Jang-Hyun Kim, Jinuk Kim, Sangwoo Kwon, Jae W. Lee, Sangdoo Yun, and Hyun Oh Song. KVzip: Query-agnostic KV cache compression with context reconstruction. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, pp. 167563–167591, 2025. arXiv:2505.23416.

Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, et al. Kimi Linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *ACM SIGOPS Symposium on Operating Systems Principles (SOSP)*, pp. 611–626, 2023. arXiv:2309.06180.

Yucheng Li, Huiqiang Jiang, Qianhui Wu, Xufang Luo, Surin Ahn, Chengruidong Zhang, Amir H. Abdi, Dongsheng Li, Jianfeng Gao, Yuqing Yang, and Lili Qiu. SCBench: A KV cache-centric analysis of long-context methods. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2412.10319.

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)*, volume 37, pp. 22947–22970, 2024. arXiv: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. *arXiv preprint arXiv:2403.19887*, 2024.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for LLM KV cache compression at test time. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 52342–52364, 2023. arXiv:2305.17118.

Weian Mao, Xi Lin, Wei Huang, Yuxin Xie, Tianfu Fu, Bohan Zhuang, Song Han, and Yukang Chen. TriAttention: Efficient long reasoning with trigonometric KV compression. In *International Conference on Machine Learning (ICML)*, 2026. arXiv:2604.04921.

MiniMax, Aonian Li, Bangwei Gong, Bo Yang, Boji Shan, Chang Liu, et al. MiniMax-01: Scaling foundation models with lightning attention. *arXiv preprint arXiv:2501.08313*, 2025.

NVIDIA. Nemotron-H: A family of accurate and efficient hybrid Mamba-Transformer models. *arXiv preprint arXiv:2504.03624*, 2025a.

NVIDIA. Nemotron-Pretraining-Dataset-sample. https://huggingface.co/datasets/nvidia/Nemotron-Pretraining-Dataset-sample, 2025b. Hugging Face dataset.

Junyoung Park, Dalton Jones, Matthew J. Morse, Raghavv Goel, Mingu Lee, and Christopher Lott. KeyDiff: Key similarity-based KV cache eviction for long-context LLM inference in resource-constrained environments. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, pp. 5983–6019, 2025. arXiv:2504.15364.

Zihan Qiu, Zekun Wang, Xiao Li, et al. On the design of Qwen3.8-Next architecture: Evaluation, efficiency, and training stability. *arXiv preprint arXiv:2608.30320*, 2026.

Qwen Team. Qwen3.8-Max: A new bar for coding and cowork. https://qwen.ai/blog?id=qwen3.8, 2026a. Blog post.

Qwen Team. Qwen3.8-27B. https://huggingface.co/Qwen/Qwen3.8-27B, 2026b. Model card and configuration.

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. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2406.07522.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning (ICML)*, volume 139 of *PMLR*, pp. 9355–9366, 2021. arXiv:2102.11174.

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. arXiv:2309.17453.

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. arXiv:2410.10819.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 115491–115522, 2024. arXiv:2406.06484.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2412.06464.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 34661–34710, 2023. arXiv:2306.14048.

A Evaluation protocol

All pruning comparisons use one deployment protocol. (1) The complete shared prefix is prefilled once into an unpruned parent session. (2) The query-agnostic prefix score of each method is computed from that parent: GateSight from the native GDN signals, SnapKV from its observation window at the end of the prefix, TriAttention from the post-normalisation keys and offline query statistics, KeyDiff from the cosine similarity of each cached key to the mean key of its head, KVzip from reconstruction attention accumulated on the parent. (3) A physically compacted child is created with `fork_session` at the requested budget; only one child is live at a time. (4) The child is reused for every question of the context, with answers token-identical to a fresh run. A context longer than the maximum length of the engine is middle-truncated to that limit before step (1), as in the official SCBench evaluation. On SCBench the two mixed tasks (`summary_with_needles`, `repoqa_and_kv`) are scored per sub-task and enter the twelve-task mean with the average of their two sub-task scores. Prefix caching is disabled, `max_num_seqs`=1, and requests run serially in one process. A result enters the main comparison only if it reports a real fork and physical cache compaction. For Qwen3.8-Flash-Next the parent snapshot holds the 36 GDN states and the model's short-convolution state, plus the 12 token-level KV page lists and the 12 block-compressed cache page lists; reuse restores all recurrent states and schedules only the question tokens. Context identity is checked with a SHA-256 of the token ids, and concurrent reuse, unknown sessions and preemption are rejected instead of attaching the wrong prefix.

Benchmarks. On SCBench every context is run with all of its questions, up to the 262K-token context limit of the models, with the official prompts and generation lengths. RULER covers eight needle-in-a-haystack variants (single needle, multi-key, multi-value and multi-query), variable tracking, common- and frequent-word extraction and two question-answering tasks, with 100 items per task generated by the official scripts (seed 42).

Metric revision. Two SCBench tasks are scored with a documented revision of the official scorer, because the official implementation mis-parses the answers it is given. (1) `qa_chn`: the official code iterates over the reference answer string, so each character is treated as a separate reference and the score is deflated; the revision normalises the prediction and every complete reference (lower case, punctuation and whitespace removed) and takes the maximum character-level F_1. On the 8,694 scored rows of Qwen3.8-27B this raises the mean by 5.6 points. (2) `choice_eng`: the official code reads the first character of the output as the option, so a response beginning with *Based on* ... is scored as option B; the revision accepts only an explicit option (`B`, `B. Beta`, `(B)`, `Option B`, `Answer: B`) and marks everything else unresolved, scoring it zero. This lowers the mean by 12.3 points. Both corrections are applied to every method, every budget and both models, and were fixed before the method rankings were inspected. The other ten tasks and the allocation ablation are unaffected. The no-prune baseline of Qwen3.8-27B moves by about the same amount on both tasks ($+6.0$ and -11.4 points), so the revision changes the metric rather than which method wins. Figures

report the revised scores, and we give the official-scorer numbers as a sensitivity analysis: under the official scorer, the twelve-task mean of every method on Qwen3.8-27B is 0.3 to 1.4 points higher at every kept fraction, and the order of the five methods at each kept fraction is unchanged.

B UTILITY LABEL

For each document the model is prefilled once with the chat prefix, the context and the assistant-turn suffix, followed by each sampled continuation ($M = 4$, up to 128 tokens each, temperature 1.0, top-p 0.95, top-k 20). For every full-attention layer the causal attention of the continuation rows over the context is recomputed in float32, and the joint leave-one-out change of Eq. 2 is evaluated in closed form with the exact cross-head metric $W_g^\top W_h$ of the output projection. The output gate is left out of the label: weighting by the gate changes the ranking within a head by almost nothing (Appendix D), and without it the entry term $v_i^\top W_g^\top W_h v_i$ depends on the entry alone, so the closed form is evaluated once per entry rather than once per (entry, continuation token) pair. Three checks run on every document: the closed form agrees with brute-force key deletion to within 1.4×10^{-5}, the hook-reconstructed module output matches the module, and every context position is covered. The label is the maximum over all continuation tokens, which rewards entries that are decisive at least once rather than mildly useful on average, and is max-normalised per (document, layer, KV head).

C DETAILS BEHIND THE OBSERVATIONS

Corpus. 2,428 documents of 750 to 1,250 tokens from nine subsets of the Nemotron pretraining sample: Nemotron-CC (high quality, high-quality synthetic, diverse QA, translated diverse QA, math), Nemotron synthetic code, and Nemotron SFT (code, general, math). Documents containing the name of any long-context benchmark are excluded.

Half-life. For a head with mean log-decay $\bar{g}$ over a document, the half-life is $\ln 0.5/\bar{g}$ tokens. The static half-life is $\ln 2/\big(\exp(A_{\log})\,\mathrm{softplus}(\mathrm{dt}_{\mathrm{bias}})\big)$, whose denominator is the value of $-g_t$ at zero input.

Sample and write-side decomposition. The analyses of §3.1 use 240 documents, every tenth of the corpus. The log effective write $\log \beta\|v\|$ of each GDN layer is split over (token, head) into a head effect, a token effect and a remainder, and the variance fractions are averaged over the 48 layers and the documents; the head effect is static, since the ordering of the 2,304 heads by mean write agrees between random document pairs at Spearman 0.96 to 0.99, as the half-life ordering does. At the 1,000-token threshold, 173 heads are slow in at least 95% of the 240 documents, 1,878 in at most 5%, and 253 in between. Slow heads also write less, at a within-layer Spearman of -0.66 between half-life and mean effective write, so the identity of a head fixes both its time scale and its typical write strength. The GDN layers therefore keep a stable division of labour, in which the token-dependent variation on top of the identity of a head separates one entry from another. On the forget side of §3.2, retention has a negative mean correlation with U (-0.12), the mirror of the positive correlation of forgetting.

Example document. The document of Figure 5 is the one whose Spearman correlation between the slow-head $\beta\|v\|$ and the KV-head mean of U at attention layer 55 is the median over the 2,428 documents; both traces are divided by their maximum over the positions after the first token.

Predictor evaluation. For every document, full-attention layer and KV head, a predictor's scores over the n context entries are compared with the raw utility U: Spearman ρ; overlap@k, the fraction of the true top-k entries that appear in the predicted top-k with $k = \max(4, \lfloor kn \rfloor)$; and coverage@$k$, the utility mass of the predicted top-k divided by that of the true top-k. Metrics are averaged over the four KV heads, then over documents, then reported per layer and as the 16-layer mean. Untrained signals are evaluated on all documents; ridge and MLP predictions are leave-one-corpus-out (each document is predicted by the model trained on the other eight corpora). The oracle single-head row selects, for each layer and KV head, the (GDN layer, head) cell with the highest mean overlap@10% on the same documents, and is therefore an in-sample ceiling for one-cell rules rather than a fair competitor. The advantage of the MLP over ridge in overlap@10% holds in all nine corpora and

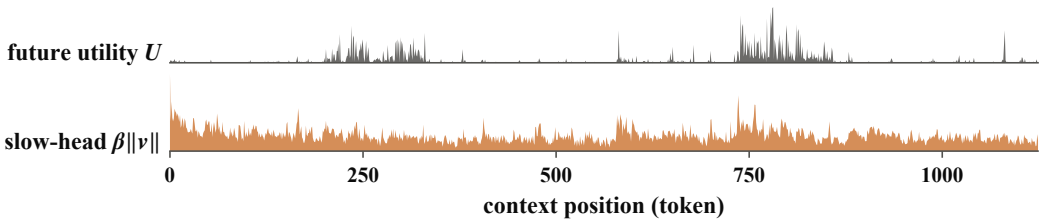

Figure 5: The example document at attention layer 55; each trace is scaled to its maximum after the first token.

is widest on code, 45.1 against 38.3% on SFT code. The learned ranking is not a position effect in disguise. The same MLP trained on the three age features alone recovers 17.9% of the top-10% entries, close to the age heuristic. Adding the GDN signals raises the overlap by 23.1 percentage points over this age-only model. Table 5 lists every untrained signal and both learned predictors with all metrics; Table 1 in the main text is its subset.

Table 5: Complete Table 1. Ov@k and Cov@k: overlap with and coverage of the true top-k% entries (%). Best in bold.

Predictor	ρ (↑)	Ov@k (↑)			Cov@k (↑)		
		5	10	20	5	10	20
Untrained signals							
random	+0.000	5.0	10.0	19.9	14.1	19.0	27.8
age (older first)	+0.071	11.6	19.2	29.4	24.4	31.2	40.0
$\|v\|$, preceding GDN layer	+0.121	7.9	13.9	24.8	19.5	24.8	34.1
β, window mean	+0.107	8.9	14.9	25.4	21.2	26.0	34.6
$-g$, window mean	+0.106	6.9	12.9	24.2	18.7	23.6	32.7
retention $\exp(g)$, window mean	−0.116	3.7	7.9	16.9	11.5	16.0	24.0
$\beta\|v\|$, window mean	+0.103	8.5	14.4	24.7	20.4	25.2	33.9
$\|v\|$ into slow heads	+0.126	9.0	15.3	25.8	21.1	26.4	35.3
slow-head $\beta\|v\|$	+0.229	10.6	18.2	30.3	24.3	30.6	40.9
best single GDN head (oracle)	+0.232	15.2	22.6	33.1	30.2	36.0	45.4
Learned predictors							
per-head MLP, 3 age features only (control)	+0.149	11.1	17.9	27.9	23.6	29.6	38.1
ridge regression, 867 features	**+0.548**	29.1	38.6	49.6	49.6	55.8	65.0
per-head MLP (GATESIGHT)	+0.543	**32.9**	**41.0**	**50.5**	**53.0**	**58.4**	**66.8**

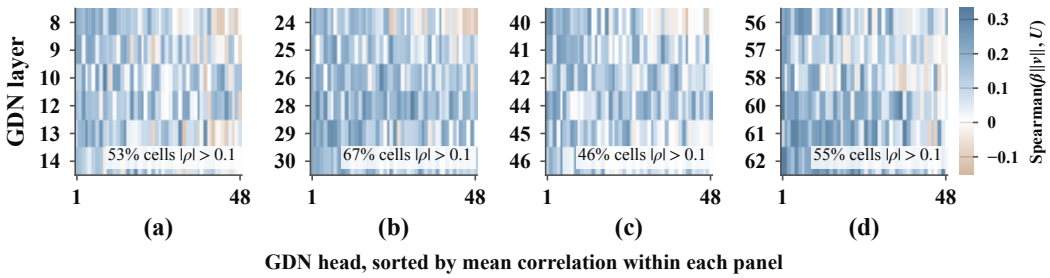

Figure 6: Per-cell correlation within the window of attention layers 15, 31, 47 and 63 (a to d).

Context length and task text. To test whether the relation of §3.2 depends on the 1K-token training corpus, we record the same utility on three further Nemotron sets whose documents are truncated to their last 2048, 4096 and 8192 tokens (40, 40 and 19 documents; the 8K set draws on three subsets only because few sample documents are that long) and on 60 RULER-8K contexts (the first 12 items of `niah_single_1`, `niah_multikey_3`, `qa_1`, `cwe` and `vt`; 7.6K to 11.1K tokens). Continuations are sampled as before. Table 6 reports the untrained signals and the deployed

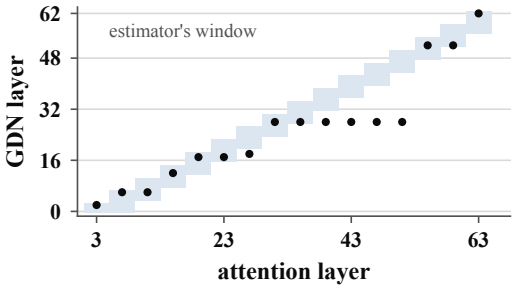

Figure 7: The GDN layer whose effective write correlates most with U (attention layers 31 to 51: layer 28; 55 and 59: layer 52).

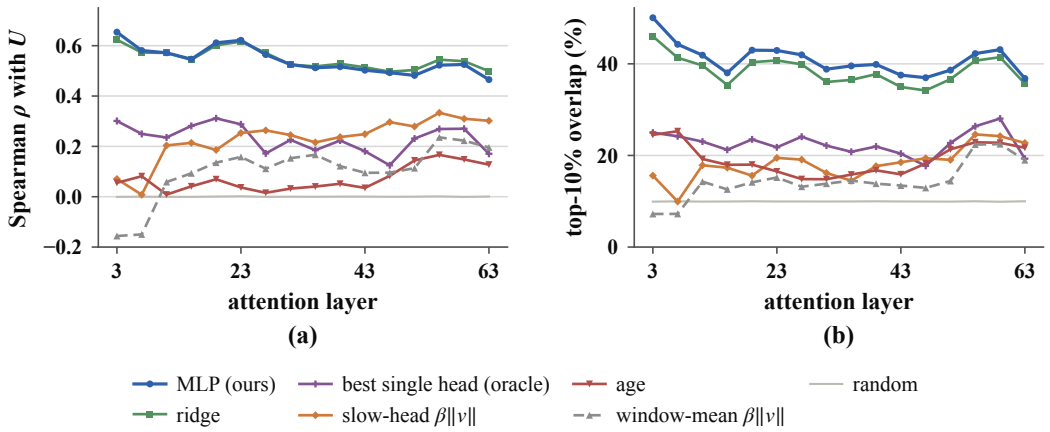

Figure 8: Per-layer version of Table 1.

estimator, the one trained on all 2,428 short documents, applied unchanged; its 1K row is therefore in-sample and only its other rows are held out. Without re-estimating the slow set, the slow-head $\beta\|v\|$ correlates with U at $+0.34$, $+0.34$ and $+0.33$ on the 2K, 4K and 8K cohorts and at $+0.30$ on the RULER-8K contexts. Per RULER task, the estimator and the slow-head write recover 42.6 and 41.1% of the top-10% entries on `niah_single_1`, 40.7 and 30.3% on `niah_multikey_3`, 31.2 and 23.4% on `qa_1`, 36.0 and 26.3% on `cwe`, and 52.3 and 45.1% on `vt`. On multi-key retrieval, where the entries to keep form spans and magnitude-based picks are scattered, the gain from learning the combination is the largest of the five task types.

Table 6: Across context length and task text: Spearman ρ with U and top-10% overlap (%). The estimator's 1K row is in-sample.

Set	Random		Age		$\beta\|v\|$, window		Slow-head $\beta\|v\|$		Estimator	
	ρ	Ov@10	ρ	Ov@10	ρ	Ov@10	ρ	Ov@10	ρ	Ov@10
Nemotron, 1K (240 docs)	+0.00	9.9	+0.09	19.3	+0.11	14.5	+0.23	18.3	+0.61	46.7
Nemotron, 2K (40)	+0.00	10.0	+0.19	29.8	+0.18	19.0	+0.34	26.3	+0.60	45.3
Nemotron, 4K (40)	+0.00	10.0	+0.10	24.4	+0.16	18.0	+0.34	27.2	+0.56	41.0
Nemotron, 8K (19)	+0.00	9.9	+0.23	31.9	+0.13	17.9	+0.33	27.8	+0.57	41.7
RULER-8K (60 contexts)	−0.00	10.0	+0.12	24.3	+0.21	25.3	+0.30	33.2	+0.43	40.6

D GATE ABLATION

The attention output gate enters GATESIGHT once, as the per-layer budget weight of §4.2. Three analyses on Qwen3.8-27B are behind that choice.

Within a head the gate does not rank tokens. With the gate applied to the value of each entry before the output projection, the ranking of the entries of a head by the attention they receive changes by almost nothing (Spearman 0.995 between the gated and the ungated scores), and end-to-end arms scored with and without the gate are indistinguishable, so the gate carries no per-token information that the estimator does not already see. The utility of Eq. 2 therefore leaves the gate out (Appendix B).

Across heads and layers it ranks strongly. The layer means of the gate span 0.022 to 0.459. Ranking the 384 query heads by their mean gate on a calibration segment and closing the 48 lowest, that is, setting their output gate to zero, costs two to four times less next-token accuracy on held-out wiki, code and math text than closing 48 random heads. At the layer level, closing the three shallowest full-attention layers outright (layers 3, 7 and 11) leaves the RULER-8K score at 96.2, tied with the unpruned model, while closing layer 23 as well collapses it to 39.1.

Per layer, not per head. With the same estimator, budgets proportional to the gate of each KV head instead of each layer score 65.9, 71.0, 74.0 and 92.0 on the RULER-8K five-task mean at $p = 0.05$, 0.1, 0.2 and 0.5, against 65.7, 71.0, 80.0 and 95.0 for the per-layer rule and 63.1, 68.9, 74.0 and 87.0 for a flat allocation: the per-head split is similar to the per-layer split at $p \leq 0.1$ and below it at $p \geq 0.2$.

Proportionality needs no exponent. On the RULER-8K analysis set at $p = 0.05$, on the tasks that remain solvable at that budget (all but multi-key retrieval), budgets proportional to the gate score 61.5, against 59.2 and 60.8 when the gate is raised to the power 0.5 or 1.5, and 53.8 for a flat allocation. The natural exponent of one is used throughout and is not treated as a hyperparameter.

E PER-TASK AND PER-BUDGET RESULTS

Figures 9 to 12 plot every task against the kept fraction for both models on both benchmarks.

SCBench. Figure 9 breaks the SCBench mean of Qwen3.8-27B into its twelve tasks. KVzip's lead at intermediate kept fractions comes from five of them. On `kv`, `repoqa_and_kv`, `prefix_suffix` and `repoqa` it leads by up to 49.6, 42.0, 16.2 and 10.7 points at $p = 0.3$ or 0.4, and on `vt` by up to 8.8 points at $p = 0.6$. `repoqa_and_kv` is the only task on which KVzip leads at every kept fraction. On the other seven tasks GATESIGHT leads on average at every $p \leq 0.3$, by 16.0 points at $p = 0.1$, with the largest leads on `summary_with_needles`, `mf` and `qa_eng` (48.3, 28.8 and 18.2 points at $p = 0.1$). From $p = 0.4$ on, the two methods differ by at most 2.9 points on these seven tasks and by less than one point on their average. At $p = 0.1$ GATESIGHT is at least as accurate as KVzip on ten of the twelve tasks, `kv`, `prefix_suffix` and `repoqa` among them. On the seven tasks it keeps 98% of the unpruned accuracy at this fraction, and KVzip 64%.

On Qwen3.8-Flash-Next (Figure 10), KVzip's lead again comes from the tasks whose answer is copied verbatim from the context. On `kv`, `prefix_suffix` and `repoqa_and_kv` it leads at every kept fraction from $p = 0.2$, by up to 88.6, 71.5 and 51.3 points at $p = 0.4$, and on `repoqa` from $p = 0.2$ to 0.7, by up to 15.8 points at $p = 0.3$. On `vt` the two methods stay within 0.4 points of each other at every kept fraction. On the other seven tasks GATESIGHT leads on average at every $p \leq 0.7$, by 23.1 points at $p = 0.1$, with the largest leads on `summary_with_needles`, `mf` and `qa_eng` (54.5, 36.5 and 33.9 points at $p = 0.1$). From $p = 0.8$ on, the two methods differ by at most 1.3 points on these seven tasks. At $p = 0.1$ GATESIGHT is at least as accurate as KVzip on eleven of the twelve tasks. On the seven tasks it keeps 98% of the unpruned accuracy at this fraction, and KVzip 51%.

RULER. Figure 11 breaks the RULER mean of Qwen3.8-27B into its thirteen tasks. GATESIGHT is at least as accurate as KVzip at every kept fraction on ten of them. On two of these, `niah_single_1` and `vt`, both methods score 100% at every kept fraction. On the other three KVzip leads at some kept fractions. On `niah_multikey_2` and `niah_multikey_3` it leads at intermediate fractions, by up to 41.0 and 42.0 points at $p = 0.2$ and 0.4. On `qa_1` it leads at every kept fraction up to 0.8, by up to 34.0 points at $p = 0.2$. At $p = 0.1$ KVzip reaches at most 2% on `niah_single_2`, `niah_single_3`, `niah_multikey_1`, `niah_multiquery` and `niah_multivalue`, and GATESIGHT at least 99.8%. These five tasks account for 81% of the

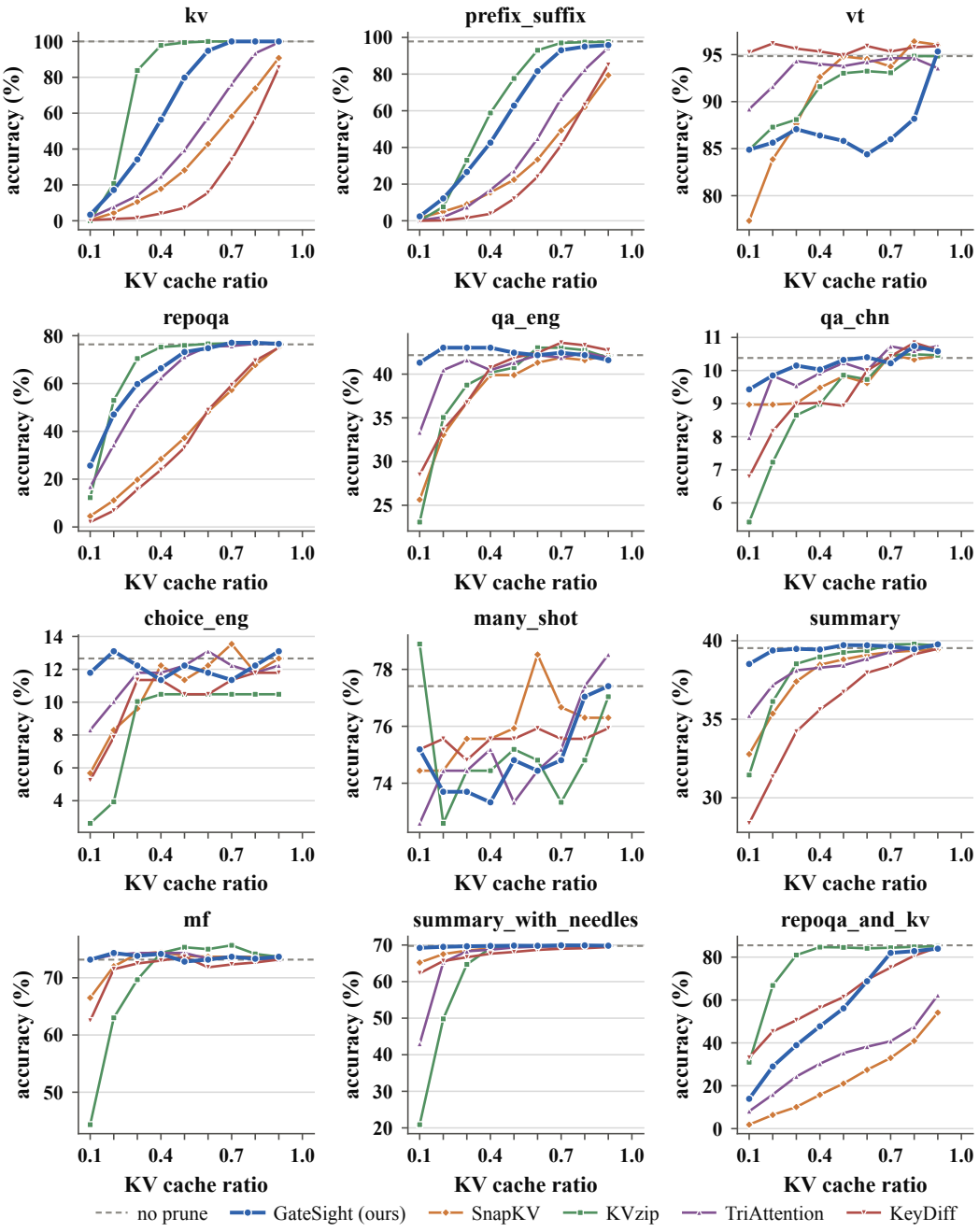

Figure 9: Per-task SCBench accuracy, Qwen3.8-27B.

47.2-point gap between the two means at $p = 0.1$. From $p = 0.6$ on, the two methods differ by at most 3.0 points on every task.

On Qwen3.8-Flash-Next (Figure 12), GATESIGHT is at least as accurate as KVzip at every kept fraction on six of the thirteen tasks: `niah_single_1`, `niah_single_2`, `niah_multivalue`, `niah_multiquery`, `vt` and `cwe`. On the other seven KVzip leads at some kept fractions. On `niah_multikey_3`, `niah_multikey_2`, `niah_single_3` and `qa_1` it leads by up to 78.0, 71.0, 67.0 and 26.0 points at $p = 0.3$ or 0.4, and on `niah_multikey_1` by up to 12.0 points at $p = 0.5$; on `fwe` and `qa_2` its lead is 1.0 point at one kept fraction. At $p = 0.1$ GATESIGHT is at least as accurate as KVzip on twelve of the thirteen tasks. From $p = 0.8$ on, the two methods differ by at most 8.0 points on every task. KeyDiff's lead on the thirteen-task mean at $p \leq 0.5$ (§5.2) comes from `niah_single_3`, `niah_multikey_1`, `niah_multivalue` and

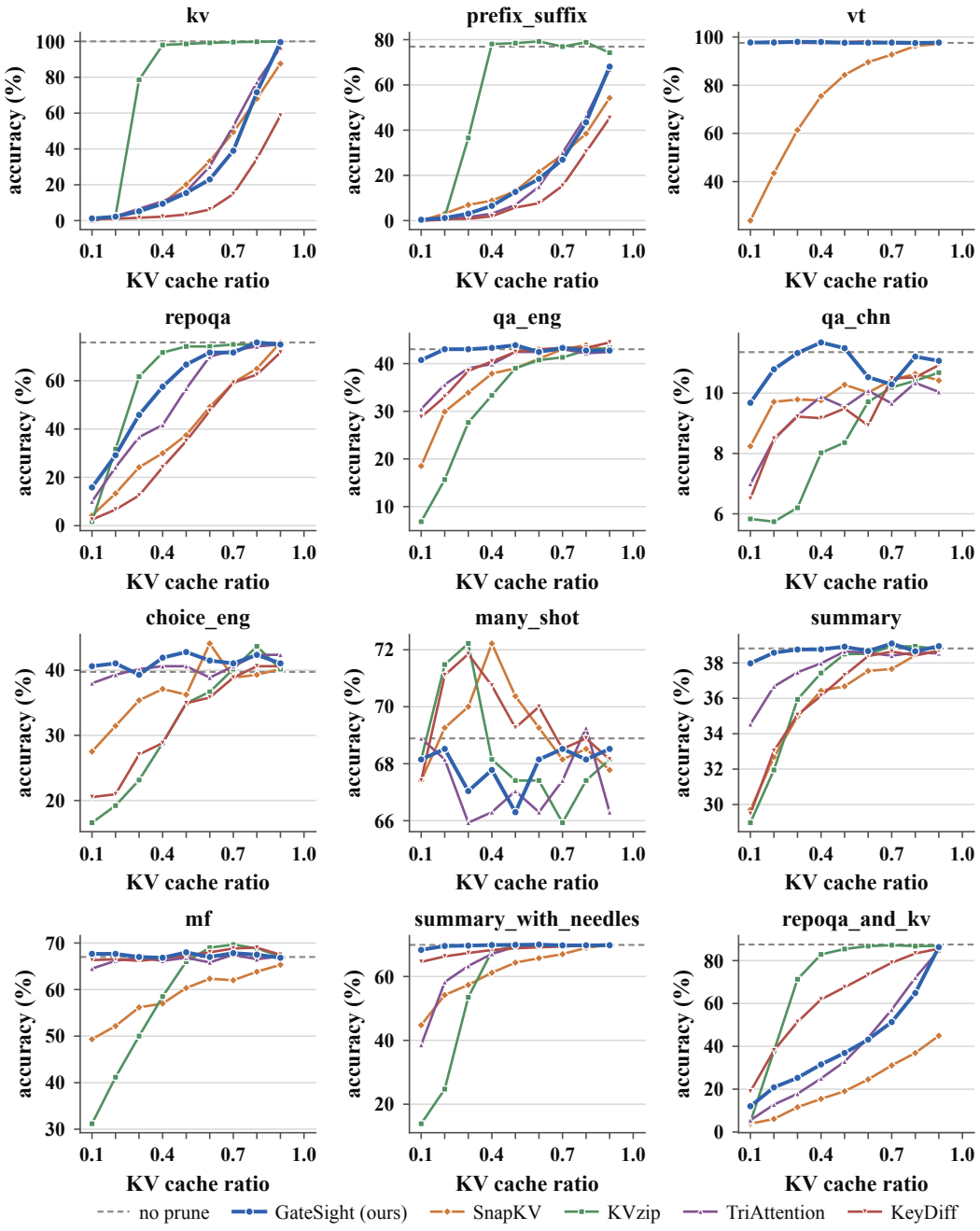

Figure 10: Per-task SCBench accuracy, Qwen3.8-Flash-Next.

`niah_multiquery`. On these four tasks KeyDiff scores at least 83.5% at every kept fraction, whereas GATESIGHT falls below 99% at every $p \leq 0.6$, and on `niah_single_3` at every $p \leq 0.7$.

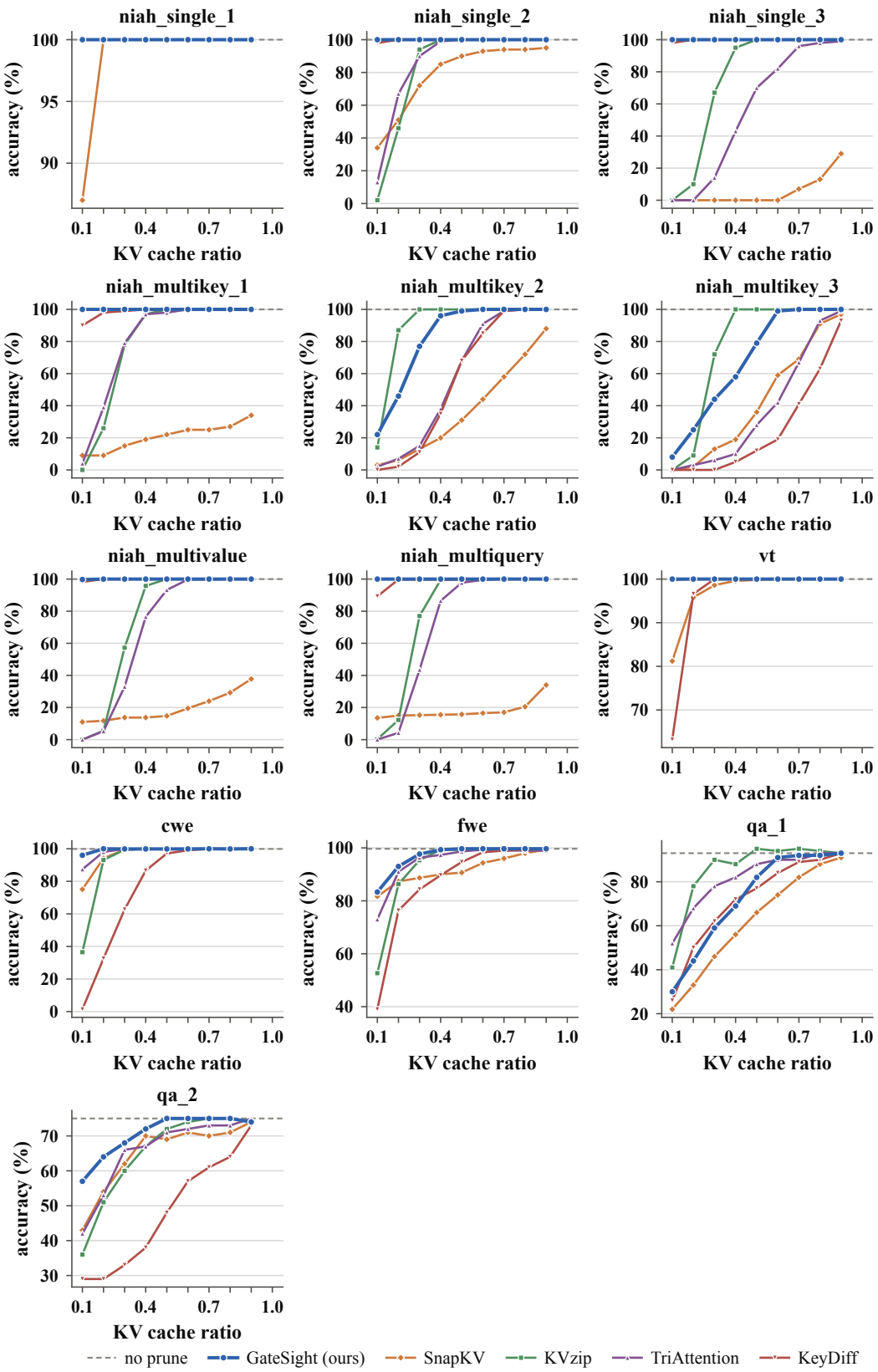

Figure 11: Per-task RULER accuracy, Qwen3.8-27B.

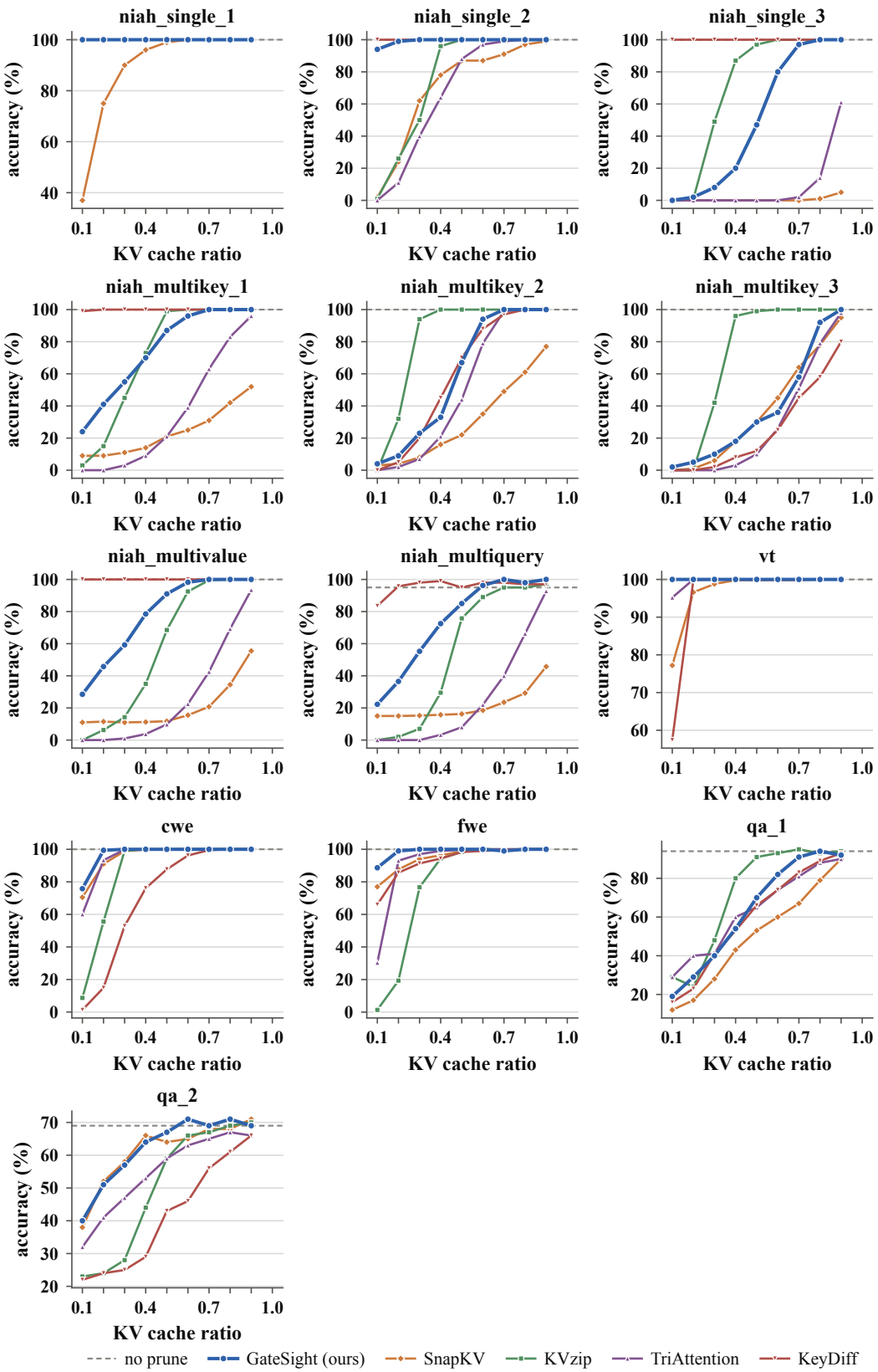

Figure 12: Per-task RULER accuracy, Qwen3.8-Flash-Next.

F Ablations

With k the flat per-layer budget, the PyramidKV schedule assigns the f-th attention layer ($f = 0, \ldots, 15$) the budget $k_{\max} - f\,(k_{\max} - k_{\min})/15$ with $k_{\min} = k/\beta$, $k_{\max} = 2k - k_{\min}$ and $\beta = 20$; the total equals the flat total, and no layer keeps fewer than four entries.

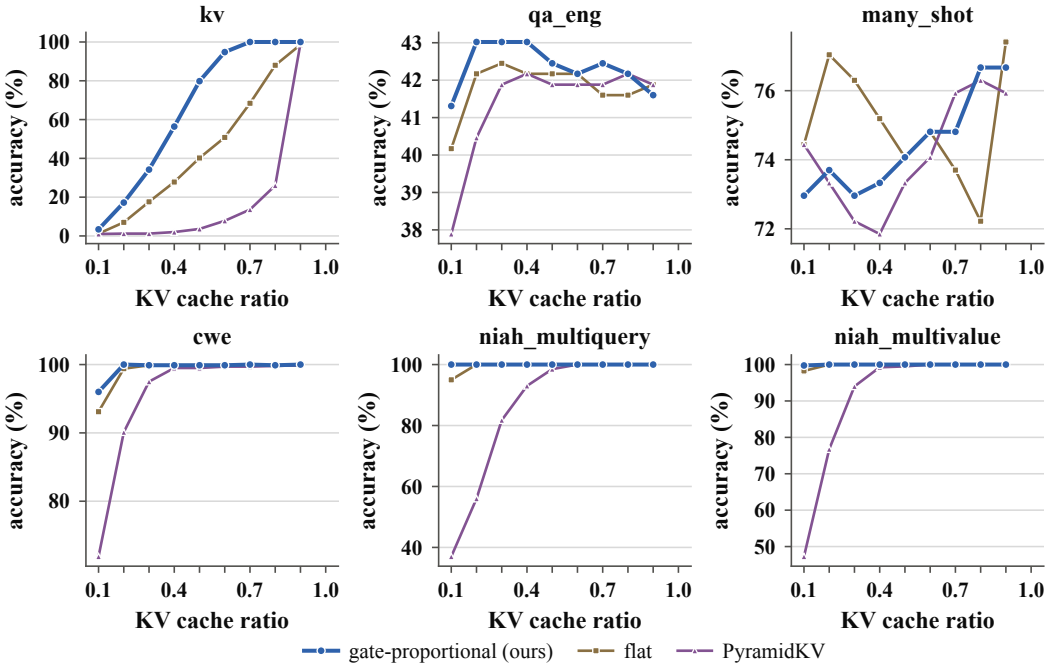

Figure 13: Allocation ablation per task, Qwen3.8-27B: SCBench (top) and RULER (bottom).

Window of GDN layers. The estimator reads the six GDN layers that precede an attention layer (§4.1). To see what this width buys, we retrain it with windows of 3, 9 and 12 layers under the shipped recipe (the same documents, label, loss, sampled positions, epochs and seed), so that only the input width changes, and evaluate as in Appendix C: leave-one-corpus-out, averaged over the 16 attention layers. As shown in Table 7, halving the window to 3 layers lowers Ov@10 by 0.8 points and ρ by 0.009. Widening it to 9 or 12 layers raises Ov@10 by 0.2 and 0.3 points and ρ by 0.004 and 0.005, for 1.5 and 1.9 times the parameters. The gain per added layer thus falls from about 0.27 points of Ov@10 below the window of 6 to 0.07 and 0.03 above it, so 6 is the last width whose signals add measurably to the prediction, and we keep it.

Table 7: Estimator window sweep (Qwen3.8-27B): held-out metrics as in Table 5, 16-layer means, with the input width and the parameter count of the 64 MLPs. Bold: the window GateSight uses.

GDN layers read	inputs	parameters	ρ	Ov@5	Ov@10	Cov@10
3	435	1.8M	+0.534	32.1	40.2	57.6
6	867	3.4M	+0.543	32.9	41.0	58.4
9	1,299	5.0M	+0.547	33.2	41.2	58.6
12	1,731	6.4M	+0.548	33.2	41.3	58.7

G Implementation details

Estimator. 64 independent MLPs $867 \to 64 \to 1$ with sigmoid output (435 inputs for the first attention layer, which has three preceding GDN layers). Binary cross-entropy against the max-normalised label, AdamW, learning rate 2×10^{-3}, weight decay 10^{-2}, gradient clipping 1.0, 12 epochs, 500 random positions per document, seed 0. Inputs are standardised with training-set statistics.

Allocation. `layer_power_budget`(gate, n_{body}, p): per-layer budget $\propto w_\ell$ (§4.2), integerised by largest remainder so that the total equals $64 \cdot \text{round}(p\,(n - \text{prot}))$; the protected prefix is kept in addition.

Serving plugin. The plugin of §4.3 patches the scheduler and the worker and gives each attention layer its own KV-cache group over the shared block pool. When prefill ends, the estimator scores the context once on the GPU; each layer then gathers its top-K_ℓ entries per head to the front of its own pages and returns the tail blocks to the pool. Since all heads of a layer keep the same count, the kept cache is a dense $K_\ell \times 4$ prefix and decoding needs only a per-layer offset. Later questions attach to the compacted cache with the GDN states restored, so only the question tokens are scheduled; one prefill can be forked into several compacted children, one per method and budget (Appendix A). For Qwen3.8-Flash-Next, whose attention layers use Qwen Sparse Attention over 4-token blocks (Qiu et al., 2026), a second plugin of the same design selects individual tokens and regroups the kept ones into new 4-token blocks (Appendix H).

Engine and compute. The Qwen3.8-27B experiments and the Flash-Next estimator training run on one NVIDIA H200 (141 GB) per job, and Qwen3.8-Flash-Next on one for RULER-4096 and on two with tensor parallelism 2 for SCBench; the Qwen3.8-27B labelling, training and pre-experiments on one NVIDIA RTX PRO 6000 Blackwell (96 GB). vLLM 0.28, PyTorch 2.13 for serving Qwen3.8-27B (Qwen3.8-Flash-Next: Appendix H); transformers 5.15.1 for labelling and training. One KV-cache group per attention layer over a shared block pool; post-prefill compaction by advanced indexing; freed blocks verified against the pool. Eager mode, prefix caching off, one sequence at a time, at most one live compacted child.

Baselines. SnapKV: observation window = last 32 context tokens, max-pooled attention, flat per-head budgets. TriAttention: trigonometric key scoring on post-RoPE keys with offline per-head query statistics calibrated on 32K Nemotron tokens; both models rotate 64 of the 256 dimensions of each head (partial rotary factor 0.25), so its positional preference is defined on a quarter of every key, and the NoPE dimensions enter as zero-frequency bands; flat per-head budgets; GQA-group max aggregation. KVzip: reconstruction attention with the paper's global top-B selection over (layer, head, token) scores, ported to the paged cache of our vLLM pipeline. KeyDiff: cosine similarity of each key, as stored in the cache after RoPE, to the unnormalised mean key of its head; the least similar keys are kept under flat per-head budgets, and the paper's block-wise streaming eviction becomes one scoring pass over the complete prefix, as for the other baselines.

H QWEN3.8-FLASH-NEXT

The runtime architecture is `Qwen4ExpForConditionalGeneration`: 48 hybrid layers, 12 attention layers with Qwen Sparse Attention over 4-token blocks, 24 query heads and 2 KV heads per attention layer, one short-convolution layer; the official FP8 checkpoint (block-wise FP8 routed experts, BF16 elsewhere); maximum context 262K tokens, the checkpoint's own limit; PLE tables offloaded to CPU. Serving uses a vLLM build at upstream commit `c5d840ff6` with a patch that offloads the PLE tables, and a second plugin of the design of §4.3.

Estimator. The estimator reads the per-token signal tensor $[36, n, 48, 3]$ and scores tokens with one MLP per (attention layer, KV head), 24 in all, each with the six-layer window and the age features of §4.1 (867 inputs, hidden 64, binary cross-entropy, AdamW, learning rate 2×10^{-3}, 8 epochs, 500 sampled tokens per document, seed 0). The training target is the utility of Eq. 2 evaluated under the sparse attention the model runs: for every continuation token, the attention of the twelve query heads of a KV head is taken over the tokens the QSA indexer selected, the entry is removed jointly from those heads with the selected set held fixed, and the changes are summed after W_O (dequantised with the FP8 checkpoint scales) before the norm; the label is the maximum over continuation tokens, max-normalised per (layer, KV head), as for Qwen3.8-27B. A token outside the selected set has zero influence on that row. Recording uses the same 2,428 Nemotron documents with four self-rollouts each, excludes the chat prefix from normalisation and training, and checks the reconstructed attention output against the native head outputs (relative L_2 error at most 5%). Training the 24 MLPs takes 12.7 minutes on one H200.

Selection and allocation. All methods select individual tokens, as on Qwen3.8-27B, but the two KV heads of a layer share one list of kept tokens; GATESIGHT ranks the tokens of a layer by the larger of its two per-KV-head predictions. The kept count of a layer is a multiple of four, and the kept tokens are regrouped into new 4-token QSA blocks whose indexer keys are pooled from the retained raw keys and rotated at the original position of the first retained token, so the native block table applies to the compacted cache. GATESIGHT allocates the token budget across the 12 attention layers in proportion to the layer mean of the output gate (the maximum over the twelve query heads of each KV head, then the mean over the two KV heads), measured once on the recording corpus of the evaluated checkpoint as in §4.2; the baselines keep flat per-layer budgets, and KVzip its global allocation, as in §5.1. On RULER-4096, replacing this allocation with a flat per-layer budget lowers the thirteen-task mean of GATESIGHT at every kept fraction, by 0.3 to 2.7 points. TriAttention statistics are calibrated on the first 32,768 rollout-context tokens of the same checkpoint.

Scoring cost. Table 8 repeats the measurement of Table 2 on Qwen3.8-Flash-Next, on two H200s with tensor parallelism 2 as in its SCBench runs. The scoring of GATESIGHT takes 1.0% to 1.4% of the time to prefill the context into a session, 0.175 s against 14.2 s at 256K tokens, and the second pass of KVzip 1.4 to 3.9 times that time.

Table 8: Wall-clock scoring time (s) per context in the same vLLM pipeline, Qwen3.8-Flash-Next, by context length; median of three runs after a warm-up.

Method	Context length (tokens)			
	4K	16K	64K	256K
GATESIGHT	0.004	0.011	0.043	0.175
TriAttention	0.011	0.011	0.013	0.030
KeyDiff	0.004	0.004	0.007	0.020
SnapKV	0.006	0.006	0.012	0.044
KVzip	0.566	1.880	8.384	55.436

I EXTENDED RELATED WORK

KV-cache eviction from observed attention. One line of work scores entries by the attention they have received. H2O keeps heavy hitters and recent tokens (Zhang et al., 2023), Scissorhands relies on the persistence of importance across decoding steps (Liu et al., 2023), SnapKV scores the prefix with an observation window at the end of the prompt (Li et al., 2024), FastGen chooses a compression policy per head from attention profiles (Ge et al., 2024), and R-KV adds a pairwise redundancy term to compress the cache of a reasoning model during decoding (Cai et al., 2025a). PyramidKV, Ada-KV and HeadKV vary the budget across layers and heads (Cai et al., 2025b; Feng et al., 2025; Fu et al., 2025). However, these methods need queries or a probe over the context, whereas GATESIGHT needs neither. StreamingLLM preserves attention sinks and a recent window (Xiao et al., 2024), whereas DuoAttention exploits a static partition into retrieval and streaming heads (Xiao et al., 2025); Observation 1 finds an analogous static structure in the recurrent layers.

Query-agnostic scoring. KVzip obtains query-agnostic scores by repeating the context and recording the attention paid during reconstruction (Kim et al., 2025); the scores are strong but cost an extra pass and an attention recompute. TriAttention replaces the probe by a trigonometric expansion of the attention score built from offline query statistics (Mao et al., 2026), which makes scoring cheap but still reads only the keys of the attention layer. KeyDiff keeps the keys least similar to the mean key of their head, a training-free geometric criterion that likewise reads only the keys (Park et al., 2025), as does the L_2 norm of keys (Devoto et al., 2024). Compared with these methods, GATESIGHT is query-agnostic like KVzip and probe-free like TriAttention and KeyDiff, but differs from all three in reading a second memory system of the model, the recurrent state, instead of the attention layer itself.

Hybrid recurrent–attention models. Mamba and its successors compress context into a state (Gu & Dao, 2024; Dao & Gu, 2024), and recent hybrids interleave such layers with attention (Lieber et al., 2024; Ren et al., 2025; NVIDIA, 2025a; De et al., 2024; MiniMax et al., 2025; Qwen Team, 2026a; Qiu et al., 2026). DeltaNet uses a data-dependent write gate (Schlag et al., 2021; Yang et al.,

2024), and Gated DeltaNet adds a data-dependent decay gate (Yang et al., 2025); these are exactly the signals we read. Our plugin keeps the paged design of vLLM (Kwon et al., 2023) and adds per-layer block tables and physical compaction. To the best of our knowledge, no prior work uses the recurrent layers of a hybrid model to manage the KV cache of its attention layers.