QUERY-AWARE KV BLOCK SELECTION FROM CACHE LANDMARKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Autoregressive language models repeatedly read a growing key-value (KV) cache when generating from long contexts. Query-aware block selection retains the cache but limits attention to blocks chosen for each new token; its quality depends on ranking those blocks well. We introduce KV-Selector[1], a small learned scorer for frozen language models that reuses the minimum and maximum key summaries stored by Quest, an earlier selection method. It trains from the model's own attention in minutes, uses fewer than 10,000 parameters per layer and adds no summaries beyond Quest's. Across eleven models, it better preserves next-token predictions than the compared training-free rules in 78 of 82 settings that restrict one layer at a time. On four models with a 640-token read budget, it lowers negative log-likelihood on PG19, a benchmark of language modelling over books, by 0.013 to 0.039 nats relative to the best training-free rule making one block choice per layer. Retrieval gains are less consistent. With the cache in host memory, selecting blocks accelerates decoding relative to synchronously copying the whole cache, and on PG19 at 131k tokens KV-Selector keeps a lower negative log-likelihood than the compared training-free rules. Our unfused implementation remains slower than full attention with the cache on the GPU.

1 INTRODUCTION

In full-attention layers of autoregressive language models, each new token reads a growing key-value (KV) cache of representations stored for previous tokens. At long context lengths, this read makes decoding expensive (Pope et al., 2023). Cache eviction discards entries after deciding which to keep (Xiao et al., 2024b; Zhang et al., 2023; Li et al., 2024). Query-aware block selection retains the cache but lets the current token's attention query read only a limited set of its blocks (Tang et al., 2024). Its quality therefore depends on choosing useful blocks for each query.

Quest, a training-free selection method, stores each block's element-wise minimum and maximum keys and uses them to bound its highest possible attention score (Tang et al., 2024). A bound can be loose, and bounding a block's best key does not by itself ensure a good ranking of blocks. Learned selectors offer another approach: SeerAttention-R trains a block gate with additional projections and cached block representations (Gao et al., 2025). This raises the question: can a small learned scorer improve block ranking using summaries already stored in the cache, while leaving the language model frozen?

We introduce KV-Selector to test this question (Figure 1). A small multilayer perceptron (MLP) combines the current query's comparisons with each block's minimum and maximum keys into one score shared by the layer's attention heads. We train the scorer using the attention the frozen model assigns to each block, without changing the language model's weights. It has fewer than 10,000 parameters per layer and needs no new projections or block summaries beyond Quest's (Section 3).

We first test block ranking by restricting one layer at a time and measuring how closely the model's next-token predictions match those of the unrestricted model. Across eleven models, including four that mix attention with other layer types, KV-Selector outperforms the compared training-free scoring rules in 78 of 82 settings (Section 4.1). On downstream benchmarks with four full-attention models and a 640-token read budget, it lowers negative log-likelihood (NLL) on PG19, a benchmark of

[1]Code: anonymous repository.

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.

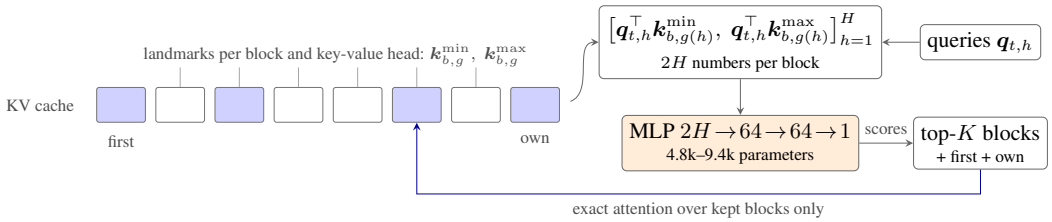

Figure 1: KV-Selector in one layer. A learned scorer ranks blocks from the query and existing block summaries. Attention reads the first block, the query's own block and the K top-ranked other blocks.

language modelling over books, by 0.013 to 0.039 nats against the best training-free rule that makes one block choice per layer (Section 4.2). Selection rules are often close on RULER, a long-context task suite; on Qwen3-8B, a rule that scores each block by its average key outperforms KV-Selector. Letting attention heads choose separate blocks narrows or reverses the PG19 margin on the two Qwen3 models.

2 Background

KV reads and block selection. An attention layer compares the current token's query vector with stored key vectors and combines the corresponding value vectors using attention weights. The keys and values form the KV cache. Block selection groups cache entries into blocks of C consecutive tokens, scores each block for the current query and reads the selected blocks (Tang et al., 2024). Attention is then computed over the selected tokens, with weights normalised over that set; its output can differ from full-cache attention. A read budget caps the tokens available to each query and layer. Larger blocks allow fewer distinct blocks to be selected at a fixed token budget.

Eviction. Eviction discards cache entries, saving memory but making them unavailable to later queries. StreamingLLM retains the first and recent tokens (Xiao et al., 2024b); H_2O also retains tokens with high accumulated attention (Zhang et al., 2023); and SnapKV keeps positions favoured by the last prompt queries (Li et al., 2024). Block selection instead keeps the whole cache and chooses for each query, so skipped blocks remain readable (Tang et al., 2024); it reduces reads, not storage.

Landmarks and fixed scores. A landmark is a compact summary vector used to score a block before reading its full keys and values. Quest (Tang et al., 2024) stores two landmarks per block b and key-value head: the element-wise minimum and maximum keys, $\boldsymbol{k}^{\min}_b, \boldsymbol{k}^{\max}_b \in \mathbb{R}^d$, where d is the head dimension. They define a box containing all keys in the block. For query $\boldsymbol{q}_h$ of head h and its key-value head's landmarks, Quest uses the largest query–key product anywhere in this box:

$$s^{\text{Quest}}_{b,h} = \sum_{i=1}^{d} \max\big(q_{h,i}\, k^{\min}_{b,i},\ q_{h,i}\, k^{\max}_{b,i}\big). \tag{1}$$

This bounds the largest actual product $\max_{j \in b} \boldsymbol{q}_h^\top \boldsymbol{k}_j$ from above. The bound is tight if one key attains every query-relevant extreme; otherwise, it can be loose. Merging blocks can increase both the bound and the largest actual product, so their difference may grow or shrink. Bound slack alone therefore does not determine ranking quality.

The Corner rule uses the same landmarks but scores a block by the larger of their two query products, $\max(\boldsymbol{q}_h^\top \boldsymbol{k}^{\min}_b, \boldsymbol{q}_h^\top \boldsymbol{k}^{\max}_b)$. The Mean-key rule uses a different summary, the block's average key $\bar{\boldsymbol{k}}_b$, and scores $\boldsymbol{q}_h^\top \bar{\boldsymbol{k}}_b$, as in MoBA and ShadowKV (Lu et al., 2025; Sun et al., 2025). These three scores are fixed functions requiring no training.

Learning to rank existing summaries. The choice of summary and the choice of scoring function are separate design decisions. Learned selectors such as SeerAttention-R train a gate using additional projections and cached block representations (Gao et al., 2025). Quest supplies compact summaries but fixes the function that ranks them. KV-Selector explores whether learning a function of the query's

two products per head with those summaries improves block selection without adding projections or another cache representation.

3 KV-SELECTOR

KV-Selector compares each query with Quest's landmarks, maps those comparisons to a score for each block, and attends to the highest-ranked blocks (Figure 1). One block choice is shared by the layer's attention heads. In this paper's default selection policy, each query reads the first block, which can attract substantial attention (Xiao et al., 2024b), its own block, and the K highest-ranked other blocks. The nominal read budget $T_{\text{read}} = (K + 2)C$ is a cap per query and layer; short prefixes, partial blocks and overlap between the first and current blocks can reduce actual reads. The scorer is trained from the frozen model's attention; only the scorer's parameters are updated.

3.1 FEATURES AND SCORER

For the query $\boldsymbol{q}_{t,h}$ of head h at position t, the scorer compares it with each block's minimum and maximum keys. With H query heads, these comparisons give $2H$ features per block:

$$\boldsymbol{x}_{t,b} = \left[\, \boldsymbol{q}_{t,h}^{\top} \boldsymbol{k}_{b,g(h)}^{\min},\ \boldsymbol{q}_{t,h}^{\top} \boldsymbol{k}_{b,g(h)}^{\max} \,\right]_{h=1}^{H} \in \mathbb{R}^{2H}. \tag{2}$$

Queries and keys are taken after rotary position encoding. Landmarks are stored per key-value head, $g(h)$ being that of query head h under grouped-query attention (Ainslie et al., 2023).

Before scoring, each feature is standardised using its mean $\boldsymbol{\mu}$ and standard deviation $\boldsymbol{\sigma}$. In the default procedure, each is computed over all query positions and blocks of the training windows and kept fixed at inference. A multilayer perceptron (MLP) specific to each layer then produces a scalar block score:

$$\tilde{\boldsymbol{x}}_{t,b} = (\boldsymbol{x}_{t,b} - \boldsymbol{\mu}) \oslash \boldsymbol{\sigma}, \qquad s_{t,b} = \mathrm{MLP}(\tilde{\boldsymbol{x}}_{t,b}), \tag{3}$$

where $\oslash$ denotes element-wise division. The MLP has two hidden layers of width 64 with GELU activations and a linear scalar output: $2H \to 64 \to 64 \to 1$. It combines both landmarks and all heads into one score, whereas the training-free rules take the maximum of their per-head scores. We did not test the two combinations separately (Section 4.1). The expanded network and parameter counts are given in Appendix A.2.

KV-Selector adds no projection of queries, keys or hidden states and uses only Quest's two landmarks per block and key-value head. With equally sized key and value arrays at matching precision, those landmarks occupy $1/C$ of the KV-cache memory, or 6.25% at $C = 16$. A fixed attention-read budget does not eliminate scoring work: the scorer still examines every candidate block's summaries.

3.2 DISTILLATION FROM BLOCK ATTENTION MASS

Training uses attention from the unrestricted frozen language model. A block's attention mass is the sum of the weights assigned to its keys by one head. Since a selected block is read whole, this mass supplies a block-level target. Taking the maximum across heads prioritises blocks important to at least one head.

Let $\alpha_{t,h,j}$ be the attention weight on key j for head h and query position t. For each candidate block, excluding the always-read first and current blocks, the target is

$$p_{t,b} \propto \max_{h} \sum_{j \in b} \alpha_{t,h,j}.$$

Each head's attention weights are normalised over all available keys before the maximum across heads is taken. The resulting scores are then normalised over candidate blocks to obtain $p_{t,b}$. The scorer minimises the cross-entropy $-\sum_b p_{t,b} \log \mathrm{softmax}_b(s_{t,b})$; only its parameters are updated.

Retaining attention mass is motivated by eviction-error bounds such as those of Feng et al. (2025). The maximum across heads used here is a heuristic for shared block selection; that analysis does not establish its optimality.

For the downstream benchmarks, the scorers train on 12 windows of 4,096 tokens from the wikitext-103 training split (Merity et al., 2017), with the language model frozen. Optimiser settings, training

recipes for the diagnostic studies and the normalisation used by each study are specified in Appendix A.

4 EXPERIMENTS

A masked-window study measures fidelity to the unrestricted model, benchmarks measure language modelling and task performance, and timing runs measure decoding cost.

Setup. The fidelity study covers eleven models (Table 5); only full-attention layers are restricted in the four hybrid models. Downstream benchmarks cover Qwen3-1.7B, Qwen3-8B, MiniCPM5-2B and K2-Horizon-3.7B. Full-attention prompt prefill predicts the first answer token and produces a cache reused at subsequent positions. Block selection is then applied at each position: RULER and LongBench generate answers token by token, while PG19 scores a continuation under teacher forcing. In both cases, the first two layers remain dense, as in Quest (Tang et al., 2024). The default block size is 16 tokens, with nominal read budgets of 640 and 2,048 tokens. The benchmark cache-reuse implementation supports only models whose layers all use full attention (Appendix A). We evaluate language modelling by NLL on 20 PG19 books at each of 8k, 16k and 32k tokens (Rae et al., 2020); synthetic long-context retrieval and reasoning by RULER at 4k–16k tokens (Hsieh et al., 2024); and downstream language tasks by six English LongBench tasks (Bai et al., 2024). Sampling and task metrics are in Appendix A.

Compared configurations. Quest, the Corner rule and the Mean-key rule are training-free: each uses the maximum score over heads to select one shared set of blocks per query and layer. The default KV-Selector learns that shared score. Its *mean-key extension* (KV-Selector + mean key in tables) adds the query's product with the block's mean key as a third feature per head; unlike the fixed Mean-key rule, it learns how to combine the features. *Per-head selection* instead gives each query head its own block choices, which can increase the total cache read. We also compare StreamingLLM, H_2O and SnapKV eviction configurations, and the released SeerAttention-R learned gates at their block size of 64.

Our StreamingLLM and H_2O adaptations advance their recent-token set during generation; H_2O's prompt-derived important set stays fixed, unlike the published policy. The eviction comparisons use attention masks over a retained cache and measure output quality rather than storage savings (Appendix B). We distinguish nominal budgets from measured reads.

Feature-statistics protocols. The main downstream benchmarks and learned-gate comparison use the fixed training statistics defined in Section 3.1. The masked-window diagnostics instead use statistics from the whole scored window, including later query positions (Section 4.1). Prompt-derived statistics were used in earlier benchmarks and the per-head scorers; the timing runs standardise each query's features over its own blocks, except one run with fixed statistics. RULER results informed the development of the normalisation procedure: the fixed-statistics default was named before its benchmark runs, but after earlier RULER evaluations. The study-specific protocols and development history are recorded in Appendix A.3 and Appendix C.10.

Intervals. Reported differences are paired over the same items within a decoding run, with 95% bootstrap intervals. These intervals describe item variation; they omit variation from repeated decoding, which moved scores by up to 0.4 points at blocks of 16 and 0.9 RULER points at blocks of 64. Retraining the benchmark scorers with two more seeds moves PG19 NLL by at most 0.001 nats and RULER by at most 0.8 points, and changes no PG19 conclusion (Table 6). For PG19, lengths can reuse the same book. Resampling whole books leaves seven of the eight comparisons against the best shared rule excluding zero; Qwen3-1.7B at 2,048 tokens has an interval of $[-0.012, +0.001]$ nats. The item-level intervals in Table 11 treat the 60 book-length pairs per model, from 20–24 books, as independent; whole-book intervals are 0.9–1.5 times as wide.

4.1 SELECTION QUALITY ACROSS ELEVEN MODELS

Fidelity measure. At a fixed read budget, useful block choices should preserve the full model's next-token distribution better than random choices. An offline attention-mass oracle provides a

Table 1: Selection fidelity with one layer restricted, budget 640 tokens. Entries average gap closed over corpus, window length, block size, restricted layer and scorer seed; random selection is 0% and the attention-mass oracle 100%. The last row gives the range over all 82 settings. Bold marks the highest model mean, not significance. Feature statistics come from the whole scored window, later queries included. Table 5 gives model and seed coverage; Table 7 gives paired leads and intervals.

Model	Layers	Cells	Quest	Corner	Mean-key	KV-Selector
Qwen3-1.7B	full	8	80	90	94	**97**
SmolLM2-1.7B	full	8	69	79	79	**90**
TinyLlama-1.1B	full	2	88	89	90	**96**
Gemma-3-1B	full + local	8	72	82	88	**89**
Qwen3-8B	full	8	57	65	69	**95**
MiniCPM5-2B	full	8	84	92	95	**96**
Granite-4.2-3B	full	8	82	83	93	**95**
K2-Horizon-3.7B	full	8	85	90	94	**96**
LFM2.5-2.6B	full + conv.	8	64	81	87	**94**
Qwen3.8-27B	full + recurrent	8	72	88	92	**94**
Muse-Glimmer-30B	full + local	8	66	81	84	**92**
All cells		82	51–94	61–97	61–98	81–99

second reference: it uses the unrestricted layer's weights to rank candidate blocks by the largest attention mass assigned by any one head. The first and query's own blocks are always read and do not compete for the K selected slots. This reference requires full-attention information that a deployed sparse selector does not compute.

With one layer, or every full-attention layer, restricted to a rule's blocks, fidelity is measured by the average next-token divergence $\mathrm{KL}_{\mathrm{rule}} = D_{\mathrm{KL}}(p_{\mathrm{full}} \,\|\, p_{\mathrm{rule}})$ over positions, where p denotes the model's next-token distribution. Gap closed is the fraction of the difference between random selection and the oracle removed by the rule:

$$\text{gap closed} = \frac{\mathrm{KL}_{\mathrm{random}} - \mathrm{KL}_{\mathrm{rule}}}{\mathrm{KL}_{\mathrm{random}} - \mathrm{KL}_{\mathrm{oracle}}}, \tag{4}$$

Random selection defines 0% and the oracle 100%. Negative values mean a larger divergence than random selection; these percentages are not task accuracy or attention recovered. The oracle need not minimise KL, so other choices can exceed 100% (Appendix C.2), as happens in some every-layer PG19 settings (Appendix C.14).

KV-Selector has the highest mean gap closed of the compared rules on all eleven models (Table 1) and leads all three training-free rules in 78 of 82 evaluation settings, a descriptive count. A setting combines corpus, window length and block size, averaged over restricted layers and scorer seeds. Each uses six evaluation windows and two or three scorer seeds (Table 5). Across settings, KV-Selector closes 81–99% of the gap, against 51–94% for Quest, 61–97% for the Corner rule and 61–98% for the Mean-key rule.

The model-level lead over the Mean-key rule, the strongest training-free comparator on each model, ranges from 1 to 26 points and is at most 2 points on five models. The reported paired intervals exclude zero on ten models; Gemma-3-1B spans $[-1.1, +3.1]$ points (Table 7). The calculation pairs layers and settings that share evaluation windows, so these are not independent replications. Another scorer seed moves a model mean by at most 0.6 points. The four settings where another rule leads, by 0.3–4.7 points, are detailed in Appendix C.1. This fidelity measure favours agreement with the attention signal used to train KV-Selector: both the training target and the oracle come from unrestricted attention. As a sensitivity check, replacing the reference with a value-norm oracle changes KV-Selector's mean gap closed on Qwen3-1.7B from 95.9 to 96.0% (Appendix C.2). This checks sensitivity to the reference, not downstream task quality. The study's feature statistics come from the whole scored window, which makes it an offline diagnostic; on Qwen3-1.7B, the only model checked, fixed statistics change gap closed by at most 0.2 points (Appendix C.10).

Table 2: Downstream scores at a nominal budget of 640 tokens. PG19 NLL is lower-is-better; RULER and LongBench percentages are higher-is-better. Q-1.7B, Q-8B, MC-2B and K2-3.7B abbreviate the four benchmark models in the setup. Bold selects numerical best scores, not significant differences; paired comparisons are in Table 11. SnapKV and learned KV-Selector + mean key use separate runs. †The latter's Qwen3-8B PG19 entry covers only the 8k and 16k books and is not ranked against the three-length entries. SnapKV has no PG19 result. The 2,048-token results are in Table 10.

	PG19 NLL ↓				RULER (%) ↑				LongBench (%) ↑			
Method	Q-1.7B	Q-8B	MC-2B	K2-3.7B	Q-1.7B	Q-8B	MC-2B	K2-3.7B	Q-1.7B	Q-8B	MC-2B	K2-3.7B
Full attention	3.125	2.552	3.063	2.747	84.5	93.9	93.1	90.7	40.5	49.5	43.8	39.2
Attention-mass oracle	3.185	2.591	3.120	2.819	82.5	92.4	92.3	89.9	40.4	49.5	43.7	40.5
Eviction (prompt-time choice)												
StreamingLLM	3.272	2.738	3.252	2.962	16.5	23.0	22.9	20.7	31.5	38.9	33.4	29.8
H_2O	3.282	2.740	3.249	2.954	26.6	29.8	31.6	30.1	33.4	41.5	37.4	31.6
SnapKV	–	–	–	–	57.2	73.1	71.8	69.5	39.1	48.0	42.8	38.6
Selection, training-free												
Quest	3.348	2.760	3.165	2.909	74.7	86.4	84.4	81.6	40.2	**49.5**	43.1	39.6
Corner rule	3.281	2.702	3.137	2.841	73.8	88.6	84.9	85.8	39.8	49.2	43.6	40.7
Mean-key rule	3.229	2.658	3.133	2.841	74.5	**88.9**	84.1	84.6	**40.2**	49.2	43.7	**40.9**
Selection, learned (ours)												
KV-Selector	**3.195**	**2.619**	3.120	2.808	**76.5**	88.0	85.2	85.7	40.2	49.5	**44.1**	40.3
KV-Selector + mean key	3.197	2.601†	**3.117**	**2.808**	76.3	88.7	**85.6**	**86.1**	40.2	49.3	43.9	40.8

With every full-attention layer of eight models restricted, KV-Selector closes 90 to 99% of the gap, above the Corner rule and Quest at both block sizes (Appendix C.5). We did not run the Mean-key rule there.

Design checks. Keeping the always-read first block matters strongly on Qwen3-1.7B and SmolLM2-1.7B; this and the reuse of choices across layers are tested in Appendix C.3.

The scorer-design tests more directly explain the learned component (Appendix C.2). On Qwen3-1.7B and SmolLM2-1.7B, a linear scorer over the same features, which also combines both landmarks and all heads, closes 2.5 and 7.0 points less of the gap than the MLP and 5.2 and 5.6 points more than the Corner rule. Training toward a hard top-K set instead of soft block attention mass costs 2.6 and 10.1 points. These tests use two seeds, eight training windows and 30 epochs, so their numerical values compare within that study. An MLP over mean-key products instead of the two landmarks closes nearly the same gap, 95.9 against 96.0% and 88.3 against 88.4%, with one stored summary per block instead of two; we did not test it downstream. The case for the min/max inputs is thus reuse of Quest's summaries, not better ranking than a learned mean-key scorer.

4.2 Long-context benchmarks

The clearest gains over shared training-free rules are on PG19; retrieval results depend more on the model and task.

On all four benchmark models, KV-Selector has lower PG19 NLL than the three shared training-free rules, StreamingLLM and H_2O at both budgets. The margin over the best shared rule is 0.013–0.039 nats at 640 tokens and 0.004–0.031 at 2,048. At 640 tokens this removes 19–37% of the best shared rule's excess NLL above full attention (Appendix C.6). In a budget sweep on Qwen3-1.7B with five scorer seeds, the Mean-key rule needs 1.3–1.8 times as many tokens for KV-Selector's NLL at 320–2,048 tokens, and Quest 6.3 times at 320 (Figure 2). Seven of the eight whole-book bootstrap intervals exclude zero; all eight do under the item-level calculation (Table 11).

Allowing separate block choices per head narrows or reverses that PG19 advantage on both Qwen3 models, but not on the other two (Table 4).

On Granite-4.2-3B, run with fewer configurations and prompt statistics, the Mean-key rule has a lower PG19 NLL than KV-Selector on the 16k and 32k books at 640 tokens (Appendix C.7).

RULER and LongBench show smaller, mixed differences. At 640 tokens, KV-Selector is 1.8 RULER points above the best shared rule on Qwen3-1.7B and shows no clear difference on MiniCPM5-2B or

Table 3: Qwen3-8B comparison at the released SeerAttention-R gates' block size of 64. PG19 covers 8k/16k only. *Our fixed blocks* uses our always-read policy. At 640 tokens, the gate's own policy reads 626–632 tokens per query versus 605–608 for the other selection configurations. The learned mean-key extension uses separate runs, whose default KV-Selector reference scores 71.3% RULER at 640 tokens, rather than 72.2% here. Bold marks numerical best scores, not significant differences.

	PG19 NLL ↓		RULER (%) ↑		LongBench (%) ↑	
Method	640	2,048	640	2,048	640	2,048
Full attention	2.547	2.547	94.0	94.0	49.5	49.5
Attention-mass oracle	2.614	2.552	91.3	93.2	49.5	49.5
Quest	2.752	2.604	74.6	90.2	46.5	49.3
Corner rule	2.705	2.592	70.3	90.7	48.2	**49.7**
Mean-key rule	**2.658**	2.584	**79.5**	**91.5**	48.4	49.5
SeerAttention-R	2.660	2.580	78.2	90.0	48.1	49.5
SeerAttention-R, our fixed blocks	2.666	2.581	76.2	90.0	48.0	49.6
KV-Selector	2.666	2.576	72.2	90.2	**48.4**	49.5
KV-Selector + mean key	2.665	**2.573**	71.3	90.4	48.3	49.1

K2-Horizon-3.7B. On Qwen3-8B it is 0.8 points below the Mean-key rule, its only benchmark-level loss at this budget with an interval excluding zero. At 2,048 tokens it loses 0.5 RULER and 0.3 LongBench points on Qwen3-8B, about the size of the observed re-decoding shift. Common-word extraction is hardest for every selection rule, 14.2 to 43.1 points below full attention, and there KV-Selector trails the best shared rule on three models (Appendix C.6). RULER's needle tasks rebuilt at 64k and 128k tokens give the same order on MiniCPM5-2B, Granite-4.2-3B and K2-Horizon-3.7B: pooled at 640 tokens, KV-Selector is 10.7 and 25.5 points above the Corner rule and Quest, with intervals excluding zero, and 2.3 points below the Mean-key rule, with interval $[-8.2, +3.3]$ (Appendix C.8).

At 640 tokens, every selection rule scores well above the tested eviction configurations on RULER, since selection retains the context that eviction discards (Table 2). In paired runs, KV-Selector leads SnapKV, which reads 660–672 tokens, by 13.4–19.3 points, and Quest leads it by 12.1–17.5. This gain does not isolate learned scoring, and the comparison excludes storage savings (Appendix C.3).

4.3 Comparison with a learned gate

We run the released SeerAttention-R decode gates for Qwen3-8B (Gao et al., 2025) at their block size of 64 tokens (Table 3). They hold about 0.92M parameters per layer, 110 times the 8,385 of KV-Selector. Their authors distilled them on mathematical reasoning traces, so we compare released artefacts rather than two designs trained on the same data. Parameter counts alone do not establish a latency or memory advantage.

The two are close on PG19 and LongBench. At 640 tokens, KV-Selector's PG19 NLL is 0.005 nats above the gate's own policy, which reads 4% more. At 2,048 tokens it is 0.004–0.005 nats below both gate configurations. No LongBench difference excludes zero. Retrieval favours the gate at 640 tokens: KV-Selector scores 72.2% on RULER, versus 76.2% with our always-read blocks and 78.2% with the gate's own policy. At 2,048 tokens there is no clear RULER difference.

Mean-key extension. At blocks of 64 and a budget of 640 tokens, the min/max landmark rules score 70.3–74.6% on RULER, versus 79.5% for the Mean-key rule. The deficits lie mostly in multi-key needle tasks (Appendix C.11). Adding the mean-key product to the learned scorer does not close the gap under the tested training recipe: in separate runs, the default and extended scorers both score 71.3%, 8.0 points below the fixed Mean-key rule.

4.4 Per-head selection

With the shared configuration's number of blocks per head, per-head Quest and KV-Selector read 1.33–2.65 times the nominal budget and gain 1.9–5.4 RULER points (Appendix C.9). Halving the

Table 4: PG19 NLL of per-head selection with half the blocks per head, minus shared KV-Selector with fixed statistics (Δ, nats, paired; *: 95% interval excludes zero). "Read": tokens read by a key-value head's query heads, relative to the shared read. Per-head KV-Selector uses prompt statistics; no per-head Mean-key rule was run.

		Shared KV-Selector	Corner rule per head		Quest per head		KV-Selector per head	
Model	Budget	NLL	read	Δ	read	Δ	read	Δ
Q-1.7B	640	3.195	0.77×	-0.003	0.73×	+0.034*	0.75×	-0.014*
Q-1.7B	2,048	3.153	0.72×	-0.015*	0.69×	-0.008*	0.70×	-0.016*
Q-8B	640	2.619	1.13×	-0.005*	0.97×	+0.021*	1.08×	-0.011*
Q-8B	2,048	2.569	1.03×	-0.004*	0.90×	-0.001	0.98×	-0.004*
MC-2B	640	3.120	1.55×	+0.067*	1.42×	+0.085*	1.40×	+0.023*
MC-2B	2,048	3.081	1.38×	+0.021*	1.20×	+0.029*	1.29×	+0.012*
K2-3.7B	640	2.808	1.17×	+0.049*	1.04×	+0.082*	1.14×	+0.024*
K2-3.7B	2,048	2.765	1.07×	+0.021*	0.94×	+0.028*	1.04×	+0.012*

blocks per head brings the read to 0.69–1.55 times the shared read (Table 4). On PG19, per-head KV-Selector and the per-head Corner rule then have a lower NLL than shared KV-Selector in all eight Qwen3 comparisons, by up to 0.016 nats with seven intervals excluding zero, at 0.70–1.13 times its read. On MiniCPM5-2B and K2-Horizon-3.7B, every per-head configuration has a higher NLL, by 0.012–0.085 nats, although all but one read more. On RULER, halving the blocks removes most of the per-head gain (Table 14); against the shared scorer with fixed statistics, the 2.8-point gain of per-head KV-Selector on Qwen3-8B shrinks to 0.3 points, with interval $[-0.4, +1.0]$. We did not time per-head selection.

4.5 Decoding speed

With the cache in GPU memory, our plain PyTorch implementation on one H200 is slower than full attention at every tested length up to 128k tokens: selection adds 18–29 ms per token (Table 19). By count, KV-Selector reads Quest's bytes, 6.7% of full attention's at 128k tokens, with 13–18% of its arithmetic against Quest's 6.7% (Table 18); we have not built a fused kernel. With the cache in host memory, it keeps landmarks on the GPU and copies only selected blocks. At 128k tokens on three models, this copies 214 times fewer bytes and decodes 8.1–34.8 times faster than synchronously copying the whole cache at each step (Figure 3, Table 20). Full attention with a GPU-resident cache is faster than both sparse offload configurations. Offloading also lowers peak GPU memory, at 128k tokens from 27.6 to 19.2 GB on K2-Horizon-3.7B. Where GPU memory lies between such peaks, full attention does not fit with its cache, and KV-Selector decodes 2.4–34.8 times faster than copying the whole cache (Table 21). With a real memory limit imposed after prefill in three such settings, full attention runs out of memory during decoding, and KV-Selector decodes 19–41 times faster than the copy (Table 22); the Corner rule does the same, faster.

These timings select in every layer and standardise each query's features over its own blocks, whereas the main benchmarks leave two layers dense and use fixed statistics. A further run times KV-Selector with fixed statistics at 128k tokens, 2.0–2.5 ms per token faster than with per-query statistics, and PG19 at 131k tokens measures quality in both layer configurations on the three timed models (Table 23, Table 24). With every layer selected and fixed statistics at 640 tokens, KV-Selector's NLL is 0.049–0.149 nats below that of the Corner rule, which copies the same bytes 1.7–7.3 ms per token faster, and 0.018–0.076 nats below that of the Mean-key rule, with every interval excluding zero; full attention is 0.11–0.13 nats lower. We did not compare with systems that prefetch or overlap copies (Sheng et al., 2023; Lee et al., 2024).

4.6 Measurement checks

Under a strict passkey diagnostic, which requires every token of the five-digit answer to be the top prediction, KV-Selector retrieves at least 97% of keys on every model, and random blocks 0–13% on seven models and 72% on Muse-Glimmer-30B (Table 9). In bfloat16, a mask that keeps every block

reaches 33 to 70% of the oracle's KL on Qwen3-1.7B and SmolLM2-1.7B with one layer restricted; measured against that mask, gap closed moves by at most 2.0 points in every checked cell, none on Qwen3-8B or TinyLlama-1.1B (Appendix C.14).

5 Related work

Query-aware block selection. Quest ranks blocks by the bound of Equation 1 over each block's landmarks (Tang et al., 2024). ArkVale scores bounding volumes and recalls evicted pages (Chen et al., 2024), and InfLLM retrieves blocks through representative tokens (Xiao et al., 2024a). SparQ and Loki rank with partial or low-rank keys (Ribar et al., 2024; Singhania et al., 2024), and RetrievalAttention, PQCache and MagicPIG index the cache with nearest-neighbour search, product quantisation and hashing (Liu et al., 2025; Zhang et al., 2025; Chen et al., 2025). Each fixes its scoring function by hand; KV-Selector keeps Quest's landmarks and the Corner rule's two dot products per head and learns only how to combine them. Landmark attention (Mohtashami & Jaggi, 2023) differs: its landmarks are trained tokens inserted into the sequence. FlexGen and InfiniGen keep the cache in host memory (Sheng et al., 2023; Lee et al., 2024), where reading fewer tokens is faster (Section 4.5).

The Mean-key rule. The Mean-key rule stores one vector per block and needs no training. It appears inside MoBA as a block gate and inside ShadowKV as a chunk score (Lu et al., 2025; Sun et al., 2025), but we have not found it as a baseline next to Quest. Of the three shared training-free rules, it is the hardest to beat in our runs, and the only one that beats KV-Selector with an interval excluding zero at blocks of 16 and 640 tokens, on RULER with Qwen3-8B (Section 4.2).

Learned selectors. Native sparse attention and its relatives train the selector with the model (Yuan et al., 2025; Lu et al., 2025; Zhao et al., 2026; DeepSeek-AI, 2025), while KV-Selector keeps it frozen. Closer to KV-Selector, SeerAttention and SeerAttention-R learn block gates (Gao et al., 2024; 2025), TokenButler, HashAttention and Spotlight Attention learn token-level predictors or hashes (Akhauri et al., 2025; Desai et al., 2025; Li et al., 2025), and AttentionPredictor and SpotAttention predict at block level (Yang et al., 2025; Ahmad & Yun, 2026). Most of these train new projections and keep a cache of their own. Steele (2026) finds that an importance scorer learned without the query does not beat hand-designed rules. KV-Selector scores every block against the current query.

Eviction and smaller caches. StreamingLLM, H_2O and SnapKV discard cache entries permanently (Xiao et al., 2024b; Zhang et al., 2023; Li et al., 2024), and FastGen picks an eviction policy per head (Ge et al., 2024). Sharing key-value heads, compressing the cache and quantising it change what is stored (Shazeer, 2019; Ainslie et al., 2023; DeepSeek-AI, 2024; Liu et al., 2024), while selection changes what a query reads. We decode the three eviction baselines at the selection budgets and record each arm's reads; our SnapKV matches a port of the reference code on 112 cases (Appendix B).

6 Conclusion

We presented KV-Selector, which replaces Quest's bound by a per-layer MLP over each head's dot products with Quest's two landmark keys, distilled from the frozen model in minutes. With one layer restricted, it has a higher mean gap closed than the three training-free rules on all eleven models, and at blocks of 16 on four benchmark models a lower PG19 NLL than every shared training-free rule, a margin that per-head rules at an approximately matched read narrow or reverse on both Qwen3 models (Section 4.2).

Selection rules mostly show no clear difference on RULER and LongBench. KV-Selector's remaining weakness is retrieval on Qwen3-8B, where the Mean-key rule beats it on RULER, at blocks of 64 even KV-Selector + mean key by 8.0 points (Section 4.3). We suspect its training data or target.

Where the cache does not fit in GPU memory, copying only the chosen blocks from host memory decodes faster than copying the whole cache on three models at 128k tokens. The Corner rule, which runs no MLP, copies the same bytes faster but has a 0.049 to 0.149 nats higher PG19 NLL at 131k tokens with every layer selected. With the cache in GPU memory, our unfused implementation is slower than full attention. Our main benchmarks stop below 128k tokens, apart from the PG19 and needle checks, and exclude hybrid models.

AI USE STATEMENT

The authors used generative AI (GenAI) tools to edit and improve text written for this paper and to draft sections of the paper. All AI-generated text was carefully reviewed prior to submission. The authors also used GenAI tools for retrieval and discovery. In addition, GenAI-assisted code support tools were employed during development. Specific implementation-related questions were also posed to chat-based GenAI models, and the responses were used as a basis for adapting the code accordingly. The authors take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 3 defines the scorer, its features, targets and training; Appendix A gives the training data, hyperparameters, seeds, protocol, software versions and hardware, and Appendix B how each baseline was implemented and checked against its reference code. All models, datasets and the SeerAttention-R gates are public. Training a scorer takes 8 to 21 minutes on one GPU. The code, the trained scorers, the per-item result files and the scripts that produce every table and figure from them are available at `https://anonymous.4open.science/r/query-aware-kv-block-selection-4F64/`, together with the float64 correctness checks of Appendix C.14.

REFERENCES

Huzama Ahmad and Se-Young Yun. SpotAttention: Plug-In Block-Sparse Routing for Pretrained Long-Context Transformers. *arXiv preprint arXiv:2606.22874*, 2026.

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)*, 2023.

Yash Akhauri, Ahmed F AbouElhamayed, Yifei Gao, Chi-Chih Chang, Sameh Gobriel, Nilesh Jain, and Mohamed S. Abdelfattah. TokenButler: Token Importance is Predictable. *arXiv preprint arXiv:2503.07518*, 2025.

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

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, et al. SmolLM2: When smol goes big – data-centric training of a small language model. *arXiv preprint arXiv:2502.02737*, 2025.

Renze Chen, Zhuofeng Wang, Beiquan Cao, Tong Wu, Size Zheng, Xiuhong Li, Xuechao Wei, Shengen Yan, Meng Li, and Yun Liang. ArkVale: Efficient generative LLM inference with recallable key-value eviction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Yang Zhou, Jianyu Zhang, Niklas Nolte, Yuandong Tian, Matthijs Douze, Leon Bottou, Zhihao Jia, and Beidi Chen. MagicPIG: LSH Sampling for Efficient LLM Generation. In *International Conference on Learning Representations (ICLR)*, 2025.

DeepSeek-AI. DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model. *arXiv preprint arXiv:2405.04434*, 2024.

DeepSeek-AI. DeepSeek-V3.2: Pushing the Frontier of Open Large Language Models. *arXiv preprint arXiv:2512.02556*, 2025.

Aditya Desai, Shuo Yang, Alejandro Cuadron, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. HashAttention: Semantic Sparsity for Faster Inference. In *International Conference on Machine Learning (ICML)*, 2025.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S. Kevin Zhou. Ada-KV: Optimizing KV Cache Eviction by Adaptive Budget Allocation for Efficient LLM Inference. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800GB dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2021.

Yizhao Gao, Zhichen Zeng, Dayou Du, Shijie Cao, Peiyuan Zhou, Jiaxing Qi, Junjie Lai, Hayden Kwok-Hay So, Ting Cao, Fan Yang, and Mao Yang. SeerAttention: Learning Intrinsic Sparse Attention in Your LLMs. *arXiv preprint arXiv:2410.13276*, 2024.

Yizhao Gao, Shuming Guo, Shijie Cao, Yuqing Xia, Yu Cheng, Lei Wang, Lingxiao Ma, Yutao Sun, Tianzhu Ye, Li Dong, Hayden Kwok-Hay So, Yu Hua, et al. SeerAttention-R: Sparse Attention Adaptation for Long Reasoning. *arXiv preprint arXiv:2506.08889*, 2025.

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.

Gemma Team. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Granite Team, IBM. Granite 4.2 language models. IBM technical blog, 2026. URL https://huggingface.co/blog/ibm-granite/granite-4-2. Released 25 August 2026; no arXiv report.

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

IFM Team. Introducing K2 Horizon: Frontier performance, radically open, 2026. URL https://ifm.ai/blog/k2/. Technical report announced for end of September 2026.

Wonbeom Lee, Jungi Lee, Junghwan Seo, and Jaewoong Sim. InfiniGen: Efficient Generative Inference of Large Language Models with Dynamic KV Cache Management. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2024.

Wenhao Li, Yuxin Zhang, Gen Luo, Haiyuan Wan, Ziyang Gong, Fei Chao, and Rongrong Ji. Spotlight Attention: Towards Efficient LLM Generation via Non-linear Hashing-based KV Cache Retrieval. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

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

Liquid AI. LFM2.5-2.6B: Agents everywhere. Liquid AI blog, 2026. URL https://www.liquid.ai/blog/lfm2-5-2-6b.

Di Liu, Meng Chen, Baotong Lu, Huiqiang Jiang, Zhenhua Han, Qianxi Zhang, Qi Chen, Chengruidong Zhang, Bailu Ding, Kai Zhang, Chen Chen, Fan Yang, et al. RetrievalAttention: Accelerating Long-Context LLM Inference via Vector Retrieval. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A Tuning-Free Asymmetric 2bit Quantization for KV Cache. In *International Conference on Machine Learning (ICML)*, 2024.

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, Zhiqi Huang, Huan Yuan, et al. MoBA: Mixture of Block Attention for Long-Context LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

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

Meta Superintelligence Lab. Muse-Glimmer-30B. Hugging Face model card, 2026. URL https://huggingface.co/meta-models/Muse-Glimmer-30B. Released August 2026; no model release report.

MiniCPM Team. MiniCPM5-2B. Hugging Face model card, 2026. URL https://huggingface.co/openbmb/MiniCPM5-2B. No separate report; the card cites the MiniCPM4 report.

Amirkeivan Mohtashami and Martin Jaggi. Landmark attention: Random-access infinite context length for transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

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. In *Conference on Machine Learning and Systems (MLSys)*, 2023.

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

Qwen Team. Qwen3.8-Max: A new bar for coding and cowork, August 2026. URL https://qwen.ai/blog?id=qwen3.8. Release announcement for the Qwen3.8 family.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations (ICLR)*, 2020. Introduces the PG-19 benchmark.

Luka Ribar, Ivan Chelombiev, Luke Hudlass-Galley, Charlie Blake, Carlo Luschi, and Douglas Orr. SparQ Attention: Bandwidth-Efficient LLM Inference. In *International Conference on Machine Learning (ICML)*, 2024.

Noam Shazeer. Fast Transformer Decoding: One Write-Head is All You Need. *arXiv preprint arXiv:1911.02150*, 2019.

Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. FlexGen: High-throughput generative inference of large language models with a single GPU. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

Prajwal Singhania, Siddharth Singh, Shwai He, Soheil Feizi, and Abhinav Bhatele. Loki: Low-rank Keys for Efficient Sparse Attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Brady Steele. On the limits of learned importance scoring for KV cache compression. *arXiv preprint arXiv:2601.14279*, 2026.

Hanshi Sun, Li-Wen Chang, Wenlei Bao, Size Zheng, Ningxin Zheng, Xin Liu, Harry Dong, Yuejie Chi, and Beidi Chen. ShadowKV: KV Cache in Shadows for High-Throughput Long-Context LLM Inference. In *International Conference on Machine Learning (ICML)*, 2025.

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

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. InfLLM: Training-Free Long-Context Extrapolation for LLMs with an Efficient Context Memory. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024a.

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)*, 2024b.

Qingyue Yang, Jie Wang, Xing Li, Zhihai Wang, Chen Chen, Lei Chen, Xianzhi Yu, Wulong Liu, Jianye Hao, Mingxuan Yuan, and Bin Li. AttentionPredictor: Temporal Patterns Matter for KV Cache Compression. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Y. X. Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, et al. Native Sparse Attention: Hardware-Aligned and Natively Trainable Sparse Attention. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025.

Hailin Zhang, Xiaodong Ji, Yilin Chen, Fangcheng Fu, Xupeng Miao, Xiaonan Nie, Weipeng Chen, and Bin Cui. PQCache: Product Quantization-based KVCache for Long Context LLM Inference. In *ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 2025.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. TinyLlama: An open-source small language model. *arXiv preprint arXiv:2401.02385*, 2024.

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)*, 2023.

Weilin Zhao, Zihan Zhou, Zhou Su, Chaojun Xiao, Yuxuan Li, Yanghao Li, Yudi Zhang, Weilun Zhao, Zhen Li, Yuxiang Huang, Ao Sun, Xu Han, et al. InfLLM-V2: Dense-Sparse Switchable Attention for Seamless Short-to-Long Adaptation. In *International Conference on Learning Representations (ICLR)*, 2026.

Table 5: The one-layer study restricts three full-attention layers per model, spread over depth, and all four on Gemma-3-1B. "Full attention" counts full-attention layers out of all layers, and "Other" names the type of the remaining layers. "Windows" gives the one-layer window lengths, where 2k, 4k and 8k are 2,048, 4,096 and 8,192 tokens. TinyLlama-1.1B, whose positions end at 2,048, is evaluated on wikitext-103 only. "Split" is the wikitext-103 split on which the one-layer scorers train, and "Seeds" the number of seeds per learned scorer in that study. "Every layer" marks the eight models of the every-layer study and the strict passkey runs. "Benchmarks" marks the four models run with every configuration and Granite-4.2-3B, run in an earlier pass with fewer configurations (Appendix C.7). Of the four, MiniCPM5-2B and K2-Horizon-3.7B joined as new families with context limits of 131k and 524k tokens. The rows follow Table 1.

Model	Full attention	Other	Restricted layers	Windows	Split	Seeds	Every layer	Benchmarks
Qwen3-1.7B	28 of 28	none	5, 14, 22	4k, 8k	validation	3	yes	every configuration
SmolLM2-1.7B	24 of 24	none	4, 12, 19	4k, 8k	validation	3	yes	no
TinyLlama-1.1B	22 of 22	none	4, 11, 17	2k	validation	3	no	no
Gemma-3-1B	4 of 26	local window	5, 11, 17, 23	4k, 8k	validation	3	no	no
Qwen3-8B	36 of 36	none	7, 18, 28	4k, 8k	validation	2	no	every configuration
MiniCPM5-2B	42 of 42	none	8, 21, 33	4k, 8k	training	3	yes	every configuration
Granite-4.2-3B	40 of 40	none	8, 20, 32	4k, 8k	training	3	yes	fewer configurations
K2-Horizon-3.7B	36 of 36	none	7, 18, 28	4k, 8k	training	3	yes	every configuration
LFM2.5-2.6B	8 of 30	convolution	5, 17, 24	4k, 8k	training	3	yes	no
Qwen3.8-27B	16 of 64	recurrent	15, 35, 51	4k, 8k	training	2	yes	no
Muse-Glimmer-30B	13 of 52	local window	11, 27, 43	4k, 8k	training	2	yes	no

A Implementation and protocol

This appendix gives the settings behind every experiment of Section 4 and the checks that ran before any result was recorded.

A.1 Models and scorer

The models are Qwen3-1.7B and Qwen3-8B (Qwen Team, 2025), SmolLM2-1.7B (Ben Allal et al., 2025), TinyLlama-1.1B (Zhang et al., 2024), Gemma-3-1B (Gemma Team, 2025), MiniCPM5-2B (MiniCPM Team, 2026), Granite-4.2-3B (Granite Team, IBM, 2026), K2-Horizon-3.7B (IFM Team, 2026), LFM2.5-2.6B (Liquid AI, 2026), Qwen3.8-27B (Qwen Team, 2026) and Muse-Glimmer-30B (Meta Superintelligence Lab, 2026).

Table 5 lists the eleven models and the experiments each ran. The other layers of the four hybrid models run unchanged, since their state does not grow with the context. Gemma-4-26B-A4B-it ran only in a reduced form (4,096 tokens, one seed, eager attention, the window placed in its answer turn after an empty thought block) and is not pooled with the eleven (Appendix C.1). We ran the models at these Hugging Face revisions, given by the first ten characters of the commit hash (full hashes in the code repository): Qwen3-1.7B `70d244cc86`, Qwen3-8B `b968826d9c`, SmolLM2-1.7B `effd688a12`, TinyLlama-1.1B `59f6f375b2`, Gemma-3-1B `dcc83ea841`, MiniCPM5-2B `12a3808a95`, Granite-4.2-3B `e459acceac`, K2-Horizon-3.7B `7c542d820f`, LFM2.5-2.6B `654f9463ce`, Qwen3.8-27B `1d4bf0f2ff`, Muse-Glimmer-30B `a4e59da52a` and Gemma-4-26B-A4B-it `4d7ae4984b`, and the SeerAttention-R gates at `b31445f675`.

A.2 Scorer architecture and training

For standardised features $\tilde{\boldsymbol{x}}_{t,b}$, the layer's scorer is

$$s_{t,b} = \boldsymbol{w}_3^\top \,\mathrm{GELU}\big(\boldsymbol{W}_2\,\mathrm{GELU}(\boldsymbol{W}_1\tilde{\boldsymbol{x}}_{t,b} + \boldsymbol{b}_1) + \boldsymbol{b}_2\big) + b_3, \qquad \tilde{\boldsymbol{x}}_{t,b} = (\boldsymbol{x}_{t,b} - \boldsymbol{\mu}) \oslash \boldsymbol{\sigma}, \tag{5}$$

The weights $\boldsymbol{W}_1, \boldsymbol{W}_2, \boldsymbol{w}_3$ and biases $\boldsymbol{b}_1, \boldsymbol{b}_2, b_3$ are learned. GELU is the Gaussian error linear unit. The network has shape $2H \to 64 \to 64 \to 1$.

The downstream benchmark scorers train on 12 windows of 4,096 tokens from the wikitext-103 training split (Merity et al., 2017), with the model frozen, for 40 epochs of Adam at learning rate 10^{-3}. The default block size is 16 tokens; the coarse-block comparison uses 64. Training the scorers of all layers takes 8 to 21 minutes per model on one GPU.

Table 6: Retraining KV-Selector with two more seeds leaves its benchmark results unchanged. KV-Selector with fixed statistics at blocks of 16, trained with scorer seeds 0, 1 and 2; seed 0 is the run reported everywhere else. "Lead" is KV-Selector minus the strongest training-free rule of the seed-0 run on the same items (PG19 in nats, lower is better; RULER in points, higher is better), paired over items and, for PG19, over whole books; * marks a 95% interval that excludes zero. The training-free rules have no seed. Seeds 1 and 2 were decoded in separate runs, so their leads also contain run-to-run variation, up to 0.4 RULER points for full attention.

Benchmark	Budget	Model	Strongest rule	KV-Selector, seeds 0 / 1 / 2	Lead, seeds 0 / 1 / 2
PG19 NLL	640	Qwen3-1.7B	Mean-key	3.195 / 3.195 / 3.196	-0.034* / -0.034* / -0.033*
PG19 NLL	640	Qwen3-8B	Mean-key	2.619 / 2.619 / 2.618	-0.039* / -0.039* / -0.040*
PG19 NLL	640	MiniCPM5-2B	Mean-key	3.120 / 3.119 / 3.119	-0.013* / -0.014* / -0.014*
PG19 NLL	640	K2-Horizon-3.7B	Mean-key	2.808 / 2.809 / 2.809	-0.032* / -0.032* / -0.031*
PG19 NLL	2048	Qwen3-1.7B	Mean-key	3.153 / 3.154 / 3.155	-0.006 / -0.005 / -0.004
PG19 NLL	2048	Qwen3-8B	Mean-key	2.569 / 2.568 / 2.568	-0.031* / -0.031* / -0.031*
PG19 NLL	2048	MiniCPM5-2B	Mean-key	3.081 / 3.080 / 3.080	-0.004* / -0.004* / -0.004*
PG19 NLL	2048	K2-Horizon-3.7B	Corner	2.765 / 2.765 / 2.765	-0.009* / -0.009* / -0.009*
RULER (%)	640	Qwen3-1.7B	Quest	76.5 / 76.1 / 76.2	+1.8* / +1.4* / +1.5*
RULER (%)	640	Qwen3-8B	Mean-key	88.0 / 88.2 / 88.0	-0.8* / -0.7 / -0.8*
RULER (%)	640	MiniCPM5-2B	Corner	85.2 / 85.2 / 85.0	+0.3 / +0.3 / +0.1
RULER (%)	640	K2-Horizon-3.7B	Corner	85.7 / 85.6 / 86.3	-0.0 / -0.2 / +0.6

For a layer with H query heads, the scorer of Equation 3 has $(2H \cdot 64 + 64) + (64 \cdot 64 + 64) + (64 + 1) = 128H + 4{,}289$ parameters. This gives 4,801 per layer on Gemma-3-1B ($H = 4$), 6,337 on Qwen3-1.7B and MiniCPM5-2B ($H = 16$), 7,361 on Qwen3.8-27B ($H = 24$), 8,385 on Qwen3-8B, SmolLM2-1.7B, TinyLlama-1.1B, K2-Horizon-3.7B, LFM2.5-2.6B and Muse-Glimmer-30B ($H = 32$), and 9,409 on Granite-4.2-3B ($H = 40$). The landmarks come from the keys as the attention layer sees them, after any rotary position encoding.

One forward pass of the frozen model over a training window yields the features (Equation 2) and targets (Section 3.2) of every layer, so all scorers of a model train on the same passes, with seed 0, which fixes their initialisation and the order of training steps; every seed uses the same training windows. For the benchmark scorers (fixed statistics, blocks of 16) we also trained seeds 1 and 2 and repeated PG19 at both budgets and RULER at 640 tokens on the same items (Table 6). Across the three seeds KV-Selector's PG19 NLL moves by at most 0.001 nats and its RULER score by at most 0.8 points. Every PG19 comparison with the strongest training-free rule reaches the same conclusion for each seed. On RULER the only change is that Qwen3-8B's deficit to the Mean-key rule excludes zero for two of the three seeds. All other results use one benchmark seed. Prompt statistics, unlike the fixed statistics stored with the weights, come from the last prompt queries of each input and stay frozen after the prefill. The scorers for blocks of 64, KV-Selector + mean key and per-head KV-Selector, which has one scorer per query head, train the same way.

A.3 Feature-statistics protocols

Fixed statistics come from training windows and remain unchanged at inference. Prompt statistics come from the last prompt queries and stay fixed during decoding. Per-query statistics, used only for timing, standardise each decoding query's features over its own blocks; no quality result uses them. One further timing run also times fixed statistics. Whole-window or whole-item statistics use all query positions and blocks of the evaluated input, including later query positions.

Study	Feature statistics
Main downstream benchmarks and learned-gate comparison	Fixed (training windows)
Earlier benchmark pass, including Granite-4.2-3B	Prompt
Shared reference for per-head comparison	Prompt
Timing runs	Per query, over its blocks; fixed in one run
Masked-window diagnostics	Whole scored window
Counting test; strict passkey (every layer)	Whole item
PG19 at 131k tokens; needle tasks at 64k and 128k	Fixed (131k PG19 also prompt)

Whole-input statistics can use future positions. The look-ahead check covered only the one-layer study on Qwen3-1.7B, not the other models (Appendix C.10). The main experimental setup states this distinction so that the diagnostic results are not read as evaluations of the fixed-statistics default.

A.4 Masked-window studies

Each evaluation window runs once with full attention and once per configuration, with one layer, or every full-attention layer, restricted to the configuration's blocks. A restricted query attends to the K chosen blocks, the first block and its own block, so at $T_{\text{read}} = 640$ a rule chooses $K = 38$ blocks at $C = 16$ and $K = 8$ at $C = 64$. We average the KL divergence of the restricted model's next-token distribution from the unrestricted model's over positions and evaluation windows, with random blocks and the oracle run on the same windows.

The one-layer study restricts the layers of Table 5 one at a time. Its scorers train on 16 windows for 60 epochs, and each cell uses 6 evaluation windows from the wikitext-103 test split or a sample of the Pile (Gao et al., 2021). The every-layer study restricts every full-attention layer at once on 4,096-token windows of held-out wikitext-103, with scorers trained on 12 windows for 40 epochs and 6 evaluation windows.

The first-block test of Section 4.1 uses the same layers of Qwen3-1.7B, SmolLM2-1.7B and Gemma-3-1B, 4,096-token windows and two seeds. When the first block competes, it is read only if it wins one of $K + 1$ chosen slots. When it is never kept, the query reads its own block and the $K + 1$ top-ranked other blocks. Every configuration of this test reads 598 to 603 tokens per query on average. To share one choice over m consecutive layers, layer l reuses the blocks chosen at layer $m\lfloor l/m \rfloor$.

A.5 Benchmarks

The prompt is prefilled with full attention into a cache that keeps every token. Each later token, generated or teacher-forced, reads its chosen blocks through an exact attention mask over the full cache, which we checked equals attention over the gathered tokens. As in Quest (Tang et al., 2024), the first two layers stay dense, and the dense prefill predicts the first answer token for every configuration, eviction included.

PG19 prefills the first $L - 2{,}048$ tokens of a test-split book of L tokens and reports the mean NLL of the last 2,048 under teacher forcing (Rae et al., 2020). The 64k check uses 10 books on MiniCPM5-2B only, since both Qwen3 models stop near 40k positions. RULER uses the pre-generated set of 13 tasks at 4k, 8k and 16k tokens with 50 items per task and length, 1,950 in all, decoded greedily in the model's chat template and scored by RULER's string-match metrics (Hsieh et al., 2024). LongBench uses 1,150 items of `narrativeqa`, `qasper`, `multifieldqa_en`, `hotpotqa`, `gov_report` and `triviaqa`, truncated in the middle to 16k tokens and scored by F1, or by ROUGE-L for `gov_report` (Bai et al., 2024).

A per-head configuration lets each query head attend only to its own top-K blocks, while the cache supplies the union over the query heads that share a key-value head. The more-read configurations keep the shared rules' K, and the approximately matched-read configurations halve it.

The 95% intervals are percentile bootstraps from 10,000 resamples at seed 0, with a book at one length as one PG19 item. We claim a difference only where its interval excludes zero.

A.6 CORRECTNESS CHECKS

Architecture tests. An adapter in our code runs the architectures of Gemma-4-26B-A4B-it and the last six models of Table 5. It handles gated queries, layers without position encoding, keys equal to values, output gates, logit post-processing and layers that are not attention. Before any GPU run, tiny random-weight models of these seven architectures and of the Qwen3, Llama and Gemma-3 families passed CPU tests in float32. The tests rebuild each attention output from its extracted inputs, check that keeping every block reproduces the logits, compare a restricted layer with attention under the same random mask, and check that the benchmark path refuses hybrid models. Each of the seven models then passed the same checks on real weights in bfloat16. A Qwen3-1.7B cell measured before the adapter existed, rerun through it on another GPU, reproduced every configuration's KL to printed precision except that of random blocks, off by 0.61% with the same blocks drawn.

Float64 gates. Before any benchmark result, each benchmark model passed five gates in float64. Scores are float64 too, and both code paths add 10^{-9} times the block index before every `torch.topk` call, so equal scores always give the same choice. G1 requires the selection code, keeping every token, to equal full attention, and G5 requires the same of the path that selects single tokens. G2 requires chunked teacher forcing to match token-by-token decoding to 10^{-4} in NLL for every configuration. G3 runs every configuration briefly. G4 requires generation from a shared prompt cache to give the greedy sequence of a fresh cache, with forced-token logits within 10^{-4}. The configurations added later, the faithful SnapKV, the SeerAttention-R gates and KV-Selector + mean key, passed them as well (Appendix C.14).

The gates caught two defects before any result existed. Float32 accumulators in index-add and index-put operations had remained in the float64 runs, and `torch.topk` broke exact ties, such as blocks with identical landmarks, by tensor shape. The second defect let a choice depend on how many queries were scored together. We restored float64 accumulation and added the tie-break above. A real fault lies far above the tolerance, since SnapKV's first implementation failed G2 at 7×10^{-2} before we fixed it.

Strict passkey metric. Each item hides a five-digit key at one of five depths in 4,096 tokens of filler, 20 items per depth. The runs use the every-layer scorers, and we did not run the Mean-key rule. The item ends with the question and the true key, so one forward pass gives the prediction for every key token given those before it. An item counts as retrieved only when every key token is the top prediction, which is the condition under which greedy decoding reproduces the key. A CPU test confirms this on 40 of 40 cases. On all eight models the full model retrieves all 100 items, and its free-running greedy generation agrees with the one-pass check on every item.

The item builder first tokenised each fragment of an item with special tokens. On MiniCPM5-2B, K2-Horizon-3.7B and Muse-Glimmer-30B, whose tokenisers add a beginning-of-sequence token, the gold key then began with that token and even the full model scored 0. The strict runs use the fixed builder, which adds special tokens only at the start of an item.

Kernel noise floor. With an explicit mask, PyTorch's scaled dot-product attention (SDPA) runs a different kernel than without one. In bfloat16, used in every experiment, even a mask that keeps every block then shifts the next-token distribution, which puts a floor under every KL we measure. The check that keeping every block reproduces the logits first ran under SDPA and failed on every model for this reason. Under eager attention it gives a KL of exactly zero, so it now runs there. Every masked-window cell of the last six models includes an configuration that keeps every block. We measured the floor for Qwen3-1.7B, SmolLM2-1.7B and Gemma-3-1B by rerunning their wikitext-103 cells, and not for Qwen3-8B or TinyLlama-1.1B. Gemma-4-26B-A4B-it runs under eager attention, since its SDPA floor reaches a KL of 5.9×10^{-2} with its five full-attention layers kept whole.

A.7 TIMING AND COMPUTE

We use PyTorch 2.5.1 and Transformers 5.15.0 with bfloat16 weights and SDPA attention, loading every model offline from a local Hugging Face cache. Runs used NVIDIA RTX 4090 (24 GB), A100 (80 GB), L40, L40S and H200 (141 GB) GPUs. All timing ran alone on one H200 in plain PyTorch.

With the cache in GPU memory we report the median milliseconds per generated token over 24 steps after 4 warm-up steps.

For offloaded decoding, an H200 node with 2 TB of host memory keeps the cache in pinned host memory. The GPU holds the landmarks, the first block and the block being filled. At $T_{\text{read}} = 640$ it scores the blocks and copies the 38 chosen blocks per layer, 608 tokens, with synchronous copies. We count the copied bytes exactly and report the median milliseconds per token. At 130,048 tokens the full copy moves 214 times as many bytes, which equals $130{,}048/608$, so every layer selects in the timed runs, including the two that stay dense in the benchmarks.

The runs on the seven adapted architectures took 55.4 GPU-hours of task time within 44.8 GPU-hours of allocation, since each worker packed several tasks onto one GPU. Of these, the one-layer study took 14.9 hours, the every-layer study 5.9 and the earlier benchmark pass 29.7 (Appendix C.7). The later runs on the four benchmark models, which added the eviction baselines, the per-head configurations, the fixed statistics, the faithful SnapKV, the SeerAttention-R head-to-head and the mean-key variant, took 238.5 GPU-hours of task time in 92 tasks within 217.5 GPU-hours of allocation, on H200, A100, L40S and L40 GPUs. The supplementary material holds the code, the trained scorers, the per-item results and the scripts that rebuild every table and figure, and the tables rebuild without a GPU.

B Baselines

Eviction. All three eviction baselines keep the full cache and apply their choice as a mask, which gives the same output as deleting the other tokens when the choice is made once after the prompt. StreamingLLM keeps the first block and the most recent blocks (Xiao et al., 2024b). Our H_2O chooses blocks once after the prompt and spends half of the budget on recent blocks and half on the blocks with the most attention accumulated over all prompt queries. The published H_2O (Zhang et al., 2023) keeps single tokens and updates its high-attention set during decoding. For these two reasons our variant may understate it on RULER, and our conclusions about eviction rest on SnapKV.

SnapKV follows the published code (Li et al., 2024). It sums the attention of the last 32 prompt queries over the prompt, averages the sums over the query heads of each key-value group and max-pools them with kernel 7. Each key-value head then keeps its top $T_{\text{read}} - 32$ positions plus the 32-query window. Its choices equal those of a port of the reference implementation on 112 random cases. The variant we ran first chose pages once per layer from a 256-query window without pooling and charged generated tokens to the budget. Only the faithful version appears in the tables (Section 5).

Pairing of the SnapKV runs. The SnapKV runs decode a batch of 9 configurations instead of 27. Because bfloat16 matrix products depend on the batch shape, the configurations that both runs contain reproduce the main run's score on 89 to 100% of RULER items and 63 to 85% of LongBench items, with means within 0.4 points. We pair each SnapKV comparison of Section 4.2 within the SnapKV runs.

SeerAttention-R. We run the released decode gates for Qwen3-8B inside our protocol at their block size of 64 (Gao et al., 2025). From the released configuration, a gate holds $128 \times 512 + 128 \times 384 = 114{,}688$ parameters per key-value head, about 0.92M per layer. Its scores match the authors' implementation on real weights to 1.2×10^{-6}, with identical block choices on 73,440 decisions. The two gate configurations differ only in the always-read blocks, taken from the gate's own rule (the current block, up to $T_{\text{read}}/C + 1$ blocks) or from ours (the first and the current block). The released cache code pools the block that holds the end of the prompt with one wrong key in layers after the first, and we did not copy this.

C Additional results

This appendix gives the results behind Section 3 and Section 4, the mean-key and two-pass tests, and the checks on the measurements.

Table 7: KV-Selector's lead in Table 1 exceeds its uncertainty on ten of the eleven models. "Lead" is KV-Selector's gap closed minus that of the model's strongest training-free rule, in points, paired over restricted layers and cells ("Units"), with a 95% bootstrap interval; * marks an interval that excludes zero. "Cells won" counts the cells in which KV-Selector beats Quest, the Corner rule and the Mean-key rule. The KL columns give the raw divergences behind gap closed, averaged over units: random blocks minus the oracle, the denominator of Equation 4, and KV-Selector minus the strongest rule.

						KL (10^{-3} nats)	
Model	Units	Strongest rule	Lead (points)	95% interval	Cells won	Random − oracle	KV-Selector − rule
Qwen3-1.7B	24	Mean-key	+2.3*	[+1.5, +3.1]	8/8	9.45	-0.21
SmolLM2-1.7B	24	Mean-key	+11.4*	[+5.9, +18.1]	8/8	11.64	-1.34
TinyLlama-1.1B	6	Mean-key	+5.1*	[+0.9, +8.5]	2/2	2.67	-0.19
Gemma-3-1B	32	Mean-key	+1.0	[-1.1, +3.1]	6/8	93.31	-0.01
Qwen3-8B	24	Mean-key	+25.6*	[+13.2, +39.1]	8/8	5.98	-1.36
MiniCPM5-2B	24	Mean-key	+0.9*	[+0.4, +1.4]	7/8	6.10	-0.08
Granite-4.2-3B	24	Mean-key	+1.8*	[+0.6, +3.0]	8/8	20.99	-0.47
K2-Horizon-3.7B	24	Mean-key	+2.8*	[+1.7, +4.2]	8/8	6.92	-0.21
LFM2.5-2.6B	24	Mean-key	+6.7*	[+5.1, +8.3]	8/8	128.42	-9.29
Qwen3.8-27B	24	Mean-key	+2.0*	[+1.0, +3.1]	7/8	24.00	-0.74
Muse-Glimmer-30B	24	Mean-key	+8.2*	[+6.2, +10.6]	8/8	3.04	-0.22

C.1 MASKED-WINDOW STUDY

A cell of Table 1 is one model, evaluation corpus, window length and block size, averaged over restricted layers and seeds. Table 7 tests KV-Selector's lead per model. Each unit is one restricted layer of one cell, and KV-Selector's gap closed, averaged over seeds, is paired with that of the model's strongest training-free rule, the Mean-key rule on all eleven models. The lead's 95% bootstrap interval excludes zero on ten models, and KV-Selector beats all three training-free rules in 78 of the 82 cells. This count is descriptive: cells share models and evaluation windows, so they are not independent tests. Units share evaluation windows across layers, so the intervals are approximate too. In raw KL, averaged over units, KV-Selector's divergence is 0.48 (Qwen3-8B) to 1.00 (Gemma-3-1B) times that of the Mean-key rule. Training with another seed moves one cell, averaged over its restricted layers, by 0.2 points at the median and 3.0 at most. The first five models contribute 34 cells and the last six 48. KV-Selector trails another rule in four cells. In two Gemma-3-1B cells on held-out wikitext-103 at blocks of 64 the Mean-key rule leads by up to 4.7 points, and in one cell each on Qwen3.8-27B and MiniCPM5-2B another rule leads by 0.3 points. Over Gemma-3-1B's wikitext-103 cells the Mean-key rule averages 88.7% against 87.3% for KV-Selector, which leads on the Pile, 90.5 against 87.1%.

Larger blocks hurt every rule, since 640 tokens buy 38 blocks of 16 but only 8 of 64. Over the 34 cells of the first five models, KV-Selector's mean gap closed falls from 96.8% at $C = 16$ to 89.0% at $C = 64$, the Corner rule's from 85.1 to 73.7% and Quest's from 76.8 to 64.5%. The Mean-key rule loses least, from 84.6 to 81.6%, and its median lead over the Corner rule grows from 1.5 to 7.7 points. KV-Selector is above Quest in all 34 cells, by a median of 19.3 points and by at least 4.3.

Most of the deficit of Quest and the Corner rule on Qwen3-8B sits at layer 7. There the two rules range from -31.8 to 26.3% over the cells, and KV-Selector from 72.2 to 98.9%. The denominator of gap closed, the KL distance between random blocks and the oracle, is small at this layer, and an independent rerun moved those cells by several points without changing the order of the rules.

A hidden-state control, a learned bilinear scorer over each block's mean hidden state (one $d \times d$ matrix per layer: 4,194,304 parameters on Qwen3-1.7B, against 6,337 for KV-Selector), averages 76.3% on the first five models. It falls to 17 to 47% on Muse-Glimmer-30B, 33 to 57% on Granite-4.2-3B and 40 to 71% on LFM2.5-2.6B, where KV-Selector closes 86 to 98%. Since the control also differs in the form of its scorer, Appendix C.2 tests the input alone.

Gemma-4-26B-A4B-it has an unrestricted loss of 6.13 nats on raw 4,096-token wikitext-103 windows, against 1.72 to 3.64 for the last six models, since its raw-text continuations degenerate into loops. With each window in its answer turn (Appendix A) the loss is 2.62 nats, and in a single-seed run KV-Selector is the best rule at both block sizes. This protocol keeps the model out of the 82 cells.

Table 8: The nonlinearity and the training target change gap closed more than richer inputs do. Mean gap closed (%) with one layer restricted, over the restricted layers, blocks of 16 and 64 and two seeds, on 4,096-token wikitext-103 windows. Parameters are per layer on Qwen3-1.7B, and GRU is a gated recurrent unit. The upper rows change the scorer's input and keep KV-Selector's target, and the lower rows keep KV-Selector's input and change the target. These scorers train on 8 windows for 30 epochs, fewer than in Table 1, so their values compare only within this table. Differences in the text are means of per-cell differences and can differ by 0.1 from differences of the rounded entries.

Scorer	Parameters	Qwen3-1.7B	SmolLM2-1.7B
Quest (training-free)	0	77.1	60.7
Corner rule (training-free)	0	88.3	75.8
Mean-key rule (training-free)	0	93.9	81.5
Linear layer over $\boldsymbol{k}_b^{\min}$, $\boldsymbol{k}_b^{\max}$ products	33	93.5	81.4
MLP over mean-key products	5,313	95.9	88.3
MLP over $\boldsymbol{k}_b^{\min}$, $\boldsymbol{k}_b^{\max}$ products (KV-Selector)	6,337	96.0	88.4
MLP over $\boldsymbol{k}_b^{\min}$, $\boldsymbol{k}_b^{\max}$ and mean-key products	7,361	96.4	89.3
Convolution over keys, with $\boldsymbol{k}_b^{\min}$, $\boldsymbol{k}_b^{\max}$ products	26,881	96.4	88.1
GRU along queries, with $\boldsymbol{k}_b^{\min}$, $\boldsymbol{k}_b^{\max}$ products	2,303,681	96.3	89.6
GRU over the layer's input hidden states	2,295,296	77.7	63.4
KV-Selector, target includes the first block	6,337	95.2	87.5
KV-Selector, peak attention instead of mass	6,337	95.7	88.6
KV-Selector, hard top-K set as target	6,337	93.4	78.3

C.2 Scorer design

We changed one part of the scorer at a time on Qwen3-1.7B and SmolLM2-1.7B (Table 8). The nonlinearity matters, since a linear layer over the same $2H$ features closes 2.5 and 7.0 points less of the gap than the MLP.

Richer key information adds little on this generic text. Mean-key products, a convolution over the block's keys and a GRU along the query positions move the mean by -0.3 to $+1.2$ points, at up to 2.3M parameters. A GRU of nearly that size over the layer's input hidden states stays near Quest, 18.6 and 26.2 points below the GRU over landmark features. We ran this pair on these two models only.

The training target matters more. Distilling toward the hard top-K set instead of the distribution of block attention mass (Section 3.2) lowers gap closed in all 6 cells on each model, by 2.6 and 10.1 points on average. Including the first block in the target costs 0.7 and 1.0 points, and peak attention instead of mass changes the mean by -0.2 and $+0.2$ points.

The attention-mass oracle does not always give the lowest KL. A value-norm oracle, which ranks blocks by the norm of their contribution to the layer output, has a lower KL in 4 of 12 one-layer cells on Qwen3-1.7B, all at layer 14, so it would close more than 100% of the gap of Equation 4 there. With it as 100%, KV-Selector's mean gap closed moves from 95.9 to 96.0%.

When we let the first block compete for one of $K + 1$ chosen slots at the same read (Section 4.1), the oracle stays at 100.0% on Qwen3-1.7B and SmolLM2-1.7B. The Corner rule falls from 88.3 and 75.8% to 66.9 and 60.7%, and the Mean-key rule from 93.9 and 81.5% to -183.1 and -426.4%, far below random blocks that keep the first block. On Gemma-3-1B, whose layers put 8 to 51% of their attention on the first block, the two settings differ by at most 0.9 points per configuration.

C.3 Selection diagnostics

Fixed-score sweep. In one sweep on layer 5 of Qwen3-1.7B, the Corner rule, which uses the same landmarks, exceeds Quest in all eight combinations of two window lengths and four block sizes. Its lead grows with C at both lengths. With 8,192-token windows, it closes 98.6% of the fidelity gap at $C = 16$ against 94.3% for Quest, and 84.4 against 71.6% at $C = 128$. This is an observed ranking result, not a consequence of monotonic bound slack.

First block and layer reuse. The default shared selection rules read the first block, which on Qwen3-1.7B and SmolLM2-1.7B draws 29 to 61% of a layer's attention on average. Negative gap closed means greater KL than the random reference, not negative accuracy. With one layer restricted, an oracle that never reads the first block closes -814.2% of the gap on Qwen3-1.7B and -343.3% on SmolLM2-1.7B, where 0% is random blocks plus the first block. Both are below the -299.7 and -271.1% of reading only the first block and the query's own block, and so are KV-Selector, the Corner rule and the Mean-key rule without the first block. On Gemma-3-1B the oracle without the first block still closes 69.8%.

When the first block instead competes for one of $K + 1$ chosen slots at the same read, the oracle stays at 100.0% on Qwen3-1.7B and SmolLM2-1.7B, but the Mean-key rule drops to -183.1 and -426.4% (Appendix C.2). KV-Selector was not run this way, since its training target excludes the first block.

One KV-Selector choice can also serve several consecutive layers. With every layer of Qwen3-1.7B restricted at blocks of 16, one choice per 7 layers closes 93.3% of the gap, against 99.3% for KV-Selector and 90.4% for the Corner rule with a choice per layer.

Eviction task breakdown. Eviction at the same budget scores far lower on retrieval. On RULER at 640 tokens, StreamingLLM scores 16.5 to 23.0% and H_2O 26.6 to 31.6%, against 76.5 to 88.0% for KV-Selector. Our SnapKV follows the published code (Li et al., 2024) and scores 57.2 to 73.1%, reading 660 to 672 tokens because it keeps generated tokens beyond the budget (Appendix B). Paired within the SnapKV runs, KV-Selector leads it by 13.4 to 19.3 points at 640 tokens and 2.4 to 9.7 at 2,048. Quest leads it by 12.1 to 17.5 and 2.1 to 9.4, and all sixteen intervals exclude zero. The comparison ignores the memory that eviction frees.

SnapKV loses most where the answer is a long exact string, returning the UUID of `niah_single_3` for 2.3% of items pooled over models, against 99.8% for KV-Selector. On LongBench, whose question ends the prompt inside SnapKV's 32-query window, SnapKV stays within 1.6 points of full attention and 1.1 to 1.9 points behind KV-Selector.

C.4 Training data and transfer

On layer 14 of Qwen3-1.7B at blocks of 64, KV-Selector closes 92.6, 94.6, 95.3, 95.7 and 95.8% of the gap with 2, 4, 8, 16 and 32 training windows, averaged over three seeds. The Corner rule closes 74.5% and Quest 45.1% on this cell, and the hidden-state control 80.4 to 86.6% at every size.

Fitted on 1,024-token windows of the same model, KV-Selector closes 92.3 to 94.9% of the gap at every length from 1,024 to 16,384 tokens, against 92.9 to 95.5% when fitted at 4,096 tokens and 67.4 to 80.9% for Quest. Past TinyLlama-1.1B's 2,048 positions its unrestricted loss reaches 4.807 nats at 4,096 tokens and 7.017 at 8,192, so gap closed there only measures agreement with a failing model.

The corpus shift of Table 1 also works in reverse. Fitted on the Pile and evaluated on held-out wikitext-103, KV-Selector is above the better of Quest and the Corner rule in all 8 cells of Qwen3-1.7B and SmolLM2-1.7B, by 4.3 to 18.2 points, and within 2.6 points of the scorer fitted on wikitext-103.

C.5 Every layer and passkey retrieval

With every full-attention layer of eight models restricted to 640 tokens on 4,096-token windows, KV-Selector stays above the Corner rule, and the Corner rule above Quest, on every model at both block sizes. KV-Selector closes 96 to 99% of the gap at blocks of 16 and 90 to 95% at 64, the Corner rule 90 to 97 and 79 to 89%, and Quest 84 to 96 and 65 to 86%. At blocks of 16, KV-Selector raises the next-token loss by 0.001 nats on Qwen3-1.7B and 0.006 on SmolLM2-1.7B, Quest by 0.017 and 0.032, and random blocks by 0.319 and 0.381. One KV-Selector choice shared by 2 consecutive layers of Qwen3-1.7B closes 98.1% of the gap.

Retrieval gives a mixed result in the same setting. We built one-needle, six-key and two-query tasks in the style of RULER inside wikitext-103 text and scored the whole answer by teacher-forced exact match. Over 18 cells on Qwen3-1.7B and SmolLM2-1.7B at budgets of 640 and 256 tokens, KV-Selector is above the better of Quest and the Corner rule in 9, level in 6 and below in 3. Its largest

Table 9: Under the strict metric, random blocks retrieve 0 to 13% of the keys on seven of the eight models, and KV-Selector retrieves at least 97% on every model. Accuracy in % over 100 items per model, each hiding a five-digit key at one of five depths in a 4,096-token prompt. Every full-attention layer reads 640 tokens in blocks of 16 or 64. KV-Selector standardises its features over all positions and blocks of the item, the answer included. The full model, the attention-mass oracle and the mask that keeps every block retrieve 100% on every model.

	Random		Quest		Corner		Hidden-state		KV-Selector	
Model	16	64	16	64	16	64	16	64	16	64
Granite-4.2-3B	0	0	100	100	100	100	100	41	100	100
K2-Horizon-3.7B	5	0	100	100	100	100	100	100	100	100
LFM2.5-2.6B	0	0	100	100	100	100	100	60	100	100
MiniCPM5-2B	1	1	100	100	100	100	100	100	100	100
Muse-Glimmer-30B	72	38	88	99	100	100	100	83	100	100
Qwen3-1.7B	0	0	100	100	100	100	98	100	100	100
Qwen3.8-27B	13	0	49	77	100	87	99	42	100	97
SmolLM2-1.7B	0	0	100	100	100	100	100	80	100	100

Table 10: At a read budget of 2,048 tokens, every selection rule is within 3.9 RULER points of full attention on each model, while StreamingLLM and H_2O score 36.2 to 56.1%. Layout, abbreviations, units, bold and † as in Table 2.

	PG19 NLL ↓				RULER (%) ↑				LongBench (%) ↑			
Method	Q-1.7B	Q-8B	MC-2B	K2-3.7B	Q-1.7B	Q-8B	MC-2B	K2-3.7B	Q-1.7B	Q-8B	MC-2B	K2-3.7B
Full attention	3.125	2.552	3.063	2.747	84.5	93.9	93.1	90.7	40.5	49.5	43.8	39.2
Attention-mass oracle	3.136	2.545	3.072	2.764	84.9	93.5	92.9	90.6	40.5	49.6	43.7	39.9
Eviction (prompt-time choice)												
StreamingLLM	3.205	2.645	3.187	2.875	36.2	40.3	41.7	40.5	35.0	42.4	37.6	32.4
H_2O	3.196	2.635	3.154	2.847	44.4	48.8	56.1	52.5	38.1	45.8	41.6	35.9
SnapKV	–	–	–	–	72.5	89.8	85.1	84.8	40.6	49.3	43.4	39.4
Selection, training-free												
Quest	3.232	2.630	3.090	2.789	81.7	92.3	91.0	86.8	40.7	49.4	43.7	40.2
Corner rule	3.181	2.606	3.087	2.774	81.1	92.6	91.3	89.1	**41.0**	**49.8**	43.7	**40.6**
Mean-key rule	3.159	2.599	3.085	2.782	80.9	**92.8**	91.5	89.0	40.3	49.6	**44.0**	40.0
Selection, learned (ours)												
KV-Selector	**3.153**	**2.569**	3.081	**2.765**	**82.1**	92.3	91.3	89.6	40.6	49.4	43.8	40.1
KV-Selector + mean key	3.154	2.556†	**3.080**	2.765	81.7	92.5	**91.8**	**89.7**	40.5	49.2	43.8	39.8

gains, 15 and 13 points, are on two-query retrieval, and its largest loss, 15 points, is on Qwen3-1.7B's six-key task at 8,192 tokens.

Table 9 gives the strict passkey runs of Section 4.6. The first-token metric gave random blocks 100% on K2-Horizon-3.7B, Muse-Glimmer-30B and Qwen3.8-27B at blocks of 16, and Quest, the Corner rule and KV-Selector 100% everywhere. Under the strict metric, KV-Selector retrieves 51 points more keys than Quest on Qwen3.8-27B at blocks of 16 (paired interval 41 to 61) and 12 more on Muse-Glimmer-30B (6 to 19). At blocks of 64 on Qwen3.8-27B it leads the Corner rule by 10 points (3 to 17).

C.6 Benchmark details

At 2,048 tokens the selection rules score 80.9 to 92.8% on RULER, against 84.5 to 93.9% for full attention and 72.5 to 89.8% for SnapKV (Table 10). Against Quest, KV-Selector lowers PG19 NLL on all four models at both budgets, by 0.010 to 0.153 nats, and it never scores below Quest on RULER or LongBench with an interval excluding zero (Table 11).

The PG19 margins look small next to the NLL but not next to what selection loses. At 640 tokens the best shared rule's NLL lies 0.070 to 0.106 nats above that of full attention, and KV-Selector removes 19 to 37% of this excess. StreamingLLM and H_2O lie 0.077 to 0.153 nats above KV-Selector, and on

Table 11: KV-Selector has a lower PG19 NLL than Quest, the Corner rule and the Mean-key rule at both budgets on all four models that ran every configuration, and on RULER and LongBench it trails the best of them with an interval excluding zero only on Qwen3-8B. Each entry is KV-Selector minus the named configuration, paired over items, or over books and lengths for PG19. PG19 is in nats, where negative favours KV-Selector, and RULER and LongBench in points, where positive favours KV-Selector. The best training-free rule is the best of Quest, the Corner rule and the Mean-key rule on that model and benchmark. * marks a 95% bootstrap interval that excludes zero.

Benchmark	Budget	Q-1.7B	Q-8B	MC-2B	K2-3.7B
PG19 NLL vs best training-free rule	640	-0.034*	-0.039*	-0.013*	-0.032*
PG19 NLL vs best training-free rule	2048	-0.006*	-0.031*	-0.004*	-0.009*
PG19 NLL vs Quest	640	-0.153*	-0.141*	-0.045*	-0.100*
PG19 NLL vs Quest	2048	-0.079*	-0.061*	-0.010*	-0.023*
RULER vs best training-free rule	640	+1.8*	-0.8*	+0.3	-0.0
RULER vs best training-free rule	2048	+0.4	-0.5*	-0.2	+0.5
RULER vs Quest	640	+1.8*	+1.7*	+0.8	+4.2*
RULER vs Quest	2048	+0.4	-0.0	+0.4	+2.8*
RULER vs SnapKV	640	+19.3*	+15.0*	+13.4*	+16.2*
RULER vs SnapKV	2048	+9.7*	+2.4*	+6.1*	+4.6*
LongBench vs best training-free rule	640	-0.0	-0.0	+0.4*	-0.6
LongBench vs best training-free rule	2048	-0.3	-0.3*	-0.1	-0.5
LongBench vs Quest	640	+0.0	-0.0	+1.1*	+0.7
LongBench vs Quest	2048	-0.1	+0.1	+0.2	-0.1
LongBench vs SnapKV	640	+1.3*	+1.5*	+1.1*	+1.9*
LongBench vs SnapKV	2048	+0.1	+0.2	+0.5*	+0.6

Qwen3-1.7B even shared Quest (3.348) is above StreamingLLM (3.272). A union configuration with half of its blocks from KV-Selector and half from the Corner rule is 0.029 nats below KV-Selector on Qwen3-1.7B, level with it on Qwen3-8B, and 0.009 and 0.010 nats above it on MiniCPM5-2B and K2-Horizon-3.7B.

We ran two further PG19 checks, one at a longer context and one over budgets. With 65,536-token prompts on MiniCPM5-2B (10 books), KV-Selector reaches 3.163 nats at 640 tokens. The shared rules lie 0.027 to 0.087 nats above it, StreamingLLM and H_2O 0.130 and 0.135, and at 2,048 tokens the shared rules lie 0.011 to 0.021 above it. A budget sweep on Qwen3-1.7B at 16k tokens (20 books) puts KV-Selector level with the oracle at 320 tokens, 3.209 against 3.211 nats, while Quest is 0.249 higher. At 4,096 tokens KV-Selector is 0.009 nats above full attention and Quest 0.070. Over five scorer seeds, KV-Selector's NLL spans at most 0.0032 nats at any budget, and every seed stays below the Mean-key rule at every budget (Figure 2). Interpolated in log budget between the measured budgets, the Mean-key rule needs 1.3 to 1.8 times as many tokens as KV-Selector to reach KV-Selector's NLL at 320 to 2,048 tokens, and Quest 6.3 times at 320; at 640 tokens and above, Quest does not reach it within 4,096.

The masked-window lead does not predict the RULER order. Qwen3-8B, where KV-Selector's one-layer lead over the Mean-key rule is largest (95 against 69%), is the only benchmark model where the Mean-key rule scores higher on RULER. At blocks of 16 no shared rule falls more than 0.7 LongBench points below full attention, and KV-Selector's 0.4-point LongBench gain on MiniCPM5-2B is about the size of the re-decoding shift.

On common-word extraction at 640 tokens, KV-Selector answers 23.9, 49.3 and 62.7% of the items on Qwen3-1.7B, Qwen3-8B and MiniCPM5-2B, against 29.3% for Quest, 58.5% for the Mean-key rule and 64.5% for the Corner rule. On K2-Horizon-3.7B it leads, 55.7 against 49.3% for Quest.

Pooled over the four models at 640 tokens, the faithful SnapKV answers 8.2% of `niah_multikey_3`, whose keys and values are UUIDs, against 50.5% for KV-Selector, and 29.4 against 47.2% of common-word extraction, while question answering is level. We suspect that keeping single positions breaks long exact strings into pieces, where a block keeps 16 consecutive tokens. On LongBench, SnapKV is 0.8 to 1.6 points below full attention at 640 tokens and at most 0.5 at 2,048, where KV-Selector leads it by 0.1 to 0.6 points, with an interval excluding zero only on

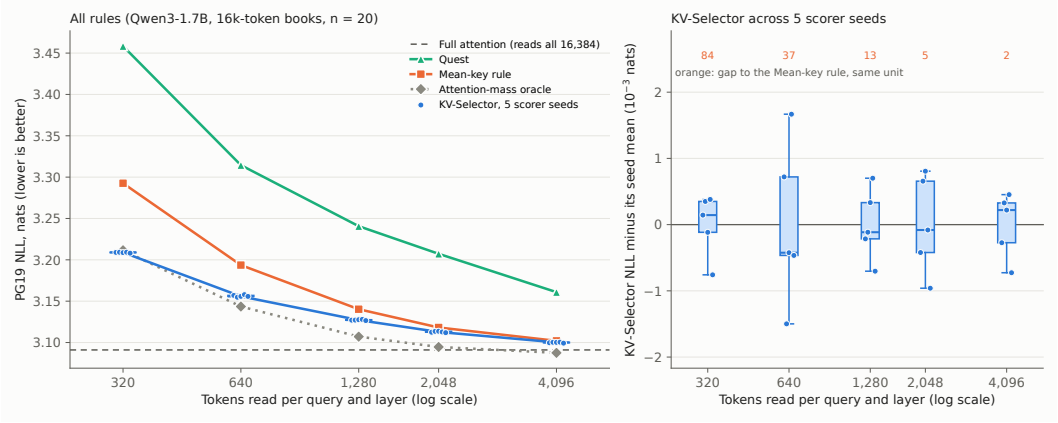

Figure 2: PG19 NLL against tokens read per query and layer on Qwen3-1.7B, 20 books of 16,384 tokens. Left: every rule, with KV-Selector (fixed statistics) for five scorer seeds. Right: each seed's NLL minus the mean over seeds, in 10^{-3} nats; the numbers above give the Mean-key rule's NLL minus KV-Selector's seed mean, in the same unit. Boxes span the middle three seeds.

Table 12: The earlier benchmark pass on Granite-4.2-3B, the model that ran with prompt statistics and without the eviction and per-head configurations. PG19 gives NLL in nats over 20 books at each length (lower is better), RULER and LongBench scores in % (higher is better). Bold marks the best selection rule in a column. The last three rows give KV-Selector minus each training-free rule, paired over items, with * for a 95% bootstrap interval that excludes zero.

	Budget 640					Budget 2048				
Arm	PG19 8k	PG19 16k	PG19 32k	RULER	LongBench	PG19 8k	PG19 16k	PG19 32k	RULER	LongBench
Full attention	2.870	2.791	2.773	88.9	39.8	2.870	2.791	2.773	88.9	39.8
Attention-mass oracle	2.905	2.848	2.849	88.3	40.1	2.869	2.796	2.783	89.0	40.1
Quest	2.951	2.926	2.948	80.1	39.5	2.885	2.829	2.838	87.1	39.7
Corner rule	2.921	2.877	2.883	81.7	39.4	**2.876**	**2.808**	2.809	87.5	39.9
Mean-key rule	2.919	**2.867**	**2.862**	83.1	**40.6**	2.876	2.812	**2.806**	87.2	40.3
KV-Selector, prompt statistics	**2.914**	2.874	2.876	**83.9**	40.1	2.879	2.816	2.817	**87.6**	**40.3**
KV-Selector − Quest	-0.037*	-0.052*	-0.072*	+3.8*	+0.6	-0.006*	-0.014*	-0.021*	+0.6	+0.6*
KV-Selector − Corner rule	-0.007*	-0.003	-0.008	+2.2*	+0.7*	+0.003*	+0.007*	+0.008*	+0.1	+0.4*
KV-Selector − Mean-key rule	-0.005	+0.007*	+0.014*	+0.8	-0.4	+0.003*	+0.003	+0.012*	+0.4	+0.0

MiniCPM5-2B. StreamingLLM and H_2O lose 6.4 to 10.6 LongBench points at 640 tokens, mostly on `multifieldqa_en` and `qasper`.

C.7 Benchmarks on Granite-4.2-3B

Granite-4.2-3B ran the benchmarks only in an earlier pass, with prompt statistics and without the eviction and per-head configurations. That pass also covered MiniCPM5-2B and K2-Horizon-3.7B, which the runs with every configuration replace, and the plan for those runs chose two new model families with long context limits over Granite-4.2-3B.

On PG19 the earlier pass goes against KV-Selector (Table 12). The Mean-key rule has a lower NLL at 640 tokens on the 16k and 32k books, by 0.007 and 0.014 nats, and the Mean-key and Corner rules both do at 2,048 tokens, by 0.003 to 0.012; those five differences have intervals excluding zero, while on the 8k books at 640 tokens KV-Selector leads the Mean-key rule by 0.005 nats with an interval including zero. Retrieval keeps the order of the other models: KV-Selector scores 83.9% on RULER at 640 tokens against 83.1 for the Mean-key rule, 81.7 for the Corner rule and 80.1 for Quest, leading Quest by 3.8 points and the Corner rule by 2.2 with intervals excluding zero, and 0.8 above the Mean-key rule with an interval including zero. On LongBench the four rules lie within 1.2 points of each other at both budgets. We did not rerun these benchmarks with fixed statistics, which moved PG19 NLL in both directions on the other models (Table 16), so we cannot tell whether they

Table 13: RULER needle tasks rebuilt at 64k and 128k tokens: % of answers found, over 25 items per row (5 tasks, 5 items each). Δ is KV-Selector minus the Mean-key rule, paired over items; * marks a 95% interval that excludes zero.

			640 tokens read					2,048 tokens read				
Model	Length	Full	KV-Selector	Mean-key	Corner	Quest	Δ	KV-Selector	Mean-key	Corner	Quest	Δ
MiniCPM5-2B	64k	96	72	72	59	46	+0	76	80	76	75	-4
	128k	75	61	57	53	24	+4	67	68	63	55	-1
Granite-4.2-3B	64k	76	62	59	51	53	+3	68	74	62	62	-6
	128k	58	38	46	33	23	-8	57	52	44	46	+5
K2-Horizon-3.7B	64k	94	62	79	54	49	-17*	80	88	79	71	-8
	128k	98	72	68	53	19	+4	84	79	74	57	+5

would change the order here. At 131k tokens, where we did, KV-Selector with fixed statistics leads the Mean-key rule on PG19 by 0.009 to 0.019 nats with intervals excluding zero, and with prompt statistics no difference from it excludes zero (Table 23). The PG19 claims of Section 4.2 cover only the four models that ran every configuration.

C.8 Needle retrieval at 64k and 128k tokens

The released RULER items stop at 16,384 tokens, so we rebuilt five of its needle tasks at 65,536 and 131,072 tokens: `niah_single_2`, `niah_multikey_1`, `niah_multikey_2`, `niah_multikey_3` and `niah_multivalue`. Instructions, needles, questions and answer prefixes follow the released items, with word keys recombined from the released needles and 7-digit or UUID values. PG19 test books that no PG19 run uses replace RULER's essays as the filler of the three tasks with a text haystack, and the two other multi-key tasks fill the context with distractor needles, as RULER does. Each task has 5 items per length, with the same keys, values and needle positions for every model, and prompts of 64.5k to 65.3k and 128.7k to 130.8k tokens. The three models are those of the timing runs; the Qwen3 models stop at 40,960 positions. The first two layers stay dense and KV-Selector uses fixed statistics, as in the benchmarks.

Pooled over the 150 items (Table 13), full attention finds 82.8% of the answers. At 640 tokens KV-Selector finds 61.2%, 10.7 points more than the Corner rule (interval $+5.7$ to $+16.0$) and 25.5 more than Quest ($+18.5$ to $+33.0$), and 2.3 fewer than the Mean-key rule (-8.2 to $+3.3$). At 2,048 tokens it finds 72.0%, 5.7 and 11.0 points more than the Corner rule and Quest, with intervals excluding zero, and 1.5 fewer than the Mean-key rule (-6.2 to $+3.0$). Of the six rows, only K2-Horizon-3.7B at 64k and 640 tokens has an interval excluding zero against the Mean-key rule, which leads there by 17 points. As on RULER at 16k, the multi-key tasks separate the rules: at 640 tokens KV-Selector answers 33.3% of `niah_multikey_2`, against 46.7% for the Mean-key rule, and no rule answers more than 6.7% of `niah_multikey_3`, whose keys and values are UUIDs.

C.9 Per-head selection

Per-head KV-Selector uses one scorer per query head, trained like the shared one. At 640 tokens and the shared rules' K, the per-head configurations of Section 4.4 read 1.33 to 2.65 times the budget and lower PG19 NLL on every model, by 0.021 to 0.177 nats. With K halved, the union read measures how far the heads of a group agree, and it falls to about half the budget when they all pick the same blocks.

On PG19 at 640 tokens this agreement largely sets the sign of the change. Per-head Quest at an approximately matched read reaches 3.229 nats on Qwen3-1.7B at 0.73 times the budget, against 3.348 for shared Quest. Against shared KV-Selector with fixed statistics, per-head KV-Selector at an approximately matched read is 0.014 and 0.011 nats lower on the two Qwen3 models and 0.023 and 0.024 higher on MiniCPM5-2B and K2-Horizon-3.7B. On RULER it loses 1.5 to 2.6 points to that reference on three models and is level on Qwen3-8B.

The per-head Corner rule, run on PG19 only, shows no clear difference from KV-Selector on Qwen3-1.7B at 640 tokens (interval -0.009 to $+0.002$ nats) and is 0.005 nats lower on Qwen3-8B at 1.13

Table 14: With half the blocks per head, per-head selection gains in 5 of the 16 comparisons and loses in 6, counting intervals that exclude zero. On PG19, per-head Quest gains on the three models whose union read stays at or below 1.04 times the budget and loses on MiniCPM5-2B, where it reads 1.42 times. "Read" is the union of tokens read by the query heads that share a key-value head, relative to the shared choice. Δ is the per-head configuration minus the shared rule, in nats for PG19 (negative is better) and in points for RULER (positive is better), and * marks a 95% interval that excludes zero. Budget 640 tokens. Per-head KV-Selector is compared with the shared scorer using prompt statistics. We did not run a per-head Mean-key rule, and ran the per-head Corner rule on PG19 only (Section 4.2).

		Quest per head		KV-Selector per head	
Benchmark	Model	read	Δ vs shared	read	Δ vs shared
PG19 NLL	Q-1.7B	0.73×	-0.120*	0.75×	+0.003
PG19 NLL	Q-8B	0.97×	-0.120*	1.08×	-0.035*
PG19 NLL	MC-2B	1.42×	+0.040*	1.40×	+0.012*
PG19 NLL	K2-3.7B	1.04×	-0.018*	1.14×	+0.013*
RULER (%)	Q-1.7B	0.70×	+0.1	0.71×	-1.2*
RULER (%)	Q-8B	0.91×	+0.0	0.99×	+2.8*
RULER (%)	MC-2B	1.22×	-3.3*	1.22×	-0.4
RULER (%)	K2-3.7B	0.91×	-0.9	0.96×	-1.9*

Table 15: At 2,048 tokens, per-head selection at an approximately matched read lowers PG19 NLL on both Qwen3 models and raises it on MiniCPM5-2B and K2-Horizon-3.7B, for Quest and KV-Selector alike. Each head chooses half as many blocks as the shared rule. Read, Δ, * and abbreviations as in Table 14, and per-head KV-Selector is again compared with the shared scorer using prompt statistics.

		Quest per head		KV-Selector per head	
Benchmark	Model	read	Δ vs shared	read	Δ vs shared
PG19 NLL	Q-1.7B	0.69×	-0.087*	0.70×	-0.015*
PG19 NLL	Q-8B	0.90×	-0.062*	0.98×	-0.019*
PG19 NLL	MC-2B	1.20×	+0.019*	1.29×	+0.008*
PG19 NLL	K2-3.7B	0.94×	+0.005*	1.04×	+0.008*
RULER (%)	Q-1.7B	0.66×	+0.4	0.69×	-0.1
RULER (%)	Q-8B	0.84×	+0.0	0.94×	+0.9*
RULER (%)	MC-2B	1.03×	-1.7*	1.13×	-1.3*
RULER (%)	K2-3.7B	0.84×	+1.9*	0.91×	-0.5

times the budget. At 2,048 tokens it is 0.004 nats lower on Qwen3-8B at 1.03 times and 0.015 lower on Qwen3-1.7B at 0.72 times, where per-head Quest is 0.008 lower at 0.69 times. On MiniCPM5-2B and K2-Horizon-3.7B, KV-Selector is lower than every per-head configuration at an approximately matched read at both budgets (Table 4).

At 2,048 tokens the sign again follows the model (Table 15). On LongBench no per-head configuration scores above its shared rule with an interval excluding zero at either budget or read, and at the approximately matched read and 640 tokens four of the eight comparisons are losses whose intervals exclude zero.

The one-layer study agrees. On layer 5 of Qwen3-1.7B at 4,096 tokens, letting each head choose lowers Quest's KL by 15% at blocks of 16 and 9% at blocks of 64, at a nearly matched read of 39.9 against 37.2 blocks per key-value head at blocks of 16. The shared Corner rule still has the lowest KL of the four configurations in both cells.

Table 16: No difference favours prompt statistics with an interval excluding zero except PG19 on Qwen3-1.7B at 640 tokens. Each entry is KV-Selector with fixed statistics minus KV-Selector with prompt statistics, paired over items, at blocks of 16. PG19 is in nats, where negative favours fixed statistics, and RULER and LongBench are in points, where positive favours fixed statistics. * marks a 95% interval that excludes zero. Abbreviations as in Table 2.

Benchmark	Budget	Q-1.7B	Q-8B	MC-2B	K2-3.7B
PG19 NLL	640	+0.017*	-0.024*	-0.011*	-0.011*
PG19 NLL	2048	+0.001	-0.015*	-0.004*	-0.003*
RULER	640	+1.4*	+2.5*	+1.2*	+0.5
RULER	2048	+1.3*	+0.9*	+0.2	+0.3
LongBench	640	-0.2	+0.4	+0.0	-0.4
LongBench	2048	+0.3*	-0.2	-0.1	+0.5*

C.10 Feature normalisation

We compute $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ on the training windows and use them unchanged at test time (fixed statistics). We first took them from the last prompt queries of each input (prompt statistics), a choice we made after seeing RULER results. We named fixed statistics the primary rule before any benchmark with them ran, but after RULER had been scored with prompt statistics on Qwen3-1.7B and Qwen3-8B. An earlier benchmark pass with prompt statistics also covered MiniCPM5-2B, K2-Horizon-3.7B and Granite-4.2-3B (Appendix C.7). It too predates the switch: those runs had started before we named the fixed rule, they carry no fixed-statistics configuration, and every benchmark with fixed statistics ran after it was named.

At 640 tokens, fixed statistics raise RULER accuracy by 0.5 to 2.5 points on all four benchmark models, with paired intervals excluding zero on three. No LongBench difference excludes zero. PG19 NLL falls on three models and rises by 0.017 nats on Qwen3-1.7B (Appendix C.10).

We compare fixed and prompt statistics (Section 3.1) on the four benchmark models in Table 16. On Qwen3-8B RULER at 640 tokens, prompt statistics put KV-Selector below all three shared rules, 3.3 points below the Mean-key rule, and fixed statistics shrink that gap to 0.8 points. At blocks of 64 they raise RULER at 640 tokens from 68.1% to the 72.2% of Table 3.

The masked-window studies take the statistics from the whole scored window, later queries included. In the one-layer study on Qwen3-1.7B this choice differs from fixed statistics by at most 0.2 points of gap closed in each of four cells. We did not check the other ten models.

C.11 Coarse blocks and the mean key

At blocks of 64 on Qwen3-8B, the rules that read only $\boldsymbol{k}_b^{\min}$ and $\boldsymbol{k}_b^{\max}$ trail the Mean-key rule on RULER at 640 tokens (Table 3). In the run of that table, pooled over the three lengths, KV-Selector's deficit sits mostly in the multi-key needle tasks, whose haystacks hold many similar key-value lines. KV-Selector answers 59.3% of `niah_multikey_2` and 16.7% of `niah_multikey_3`, against 94.7 and 51.3% for the Mean-key rule and 100.0% for full attention.

Among the landmark rules, the order of the masked-window study reverses at this block size. Quest scores 2.4 points above KV-Selector and 4.3 above the Corner rule, mostly on variable tracking (94.0 against 66.3% for KV-Selector) and multi-value needles (82.2 against 68.0%). This is Quest's only lead over KV-Selector with an interval excluding zero in any benchmark setting, so at coarse blocks the looseness of its bound does not predict retrieval. On PG19 at 640 tokens, the Mean-key rule is 0.008 nats and the gate's own rule 0.005 nats below KV-Selector, both with intervals excluding zero, while the gate given our always-read blocks equals KV-Selector.

KV-Selector + mean key (Section 4.3) adds the query's product with each block's mean key as a third feature per head, with $192H + 4{,}289$ parameters per layer: 10,433 at 32 heads against 8,385. Its third landmark raises the landmark memory from $2/C$ to $3/C$ of the keys. We trained it like KV-Selector, with fixed statistics. Its runs also decode KV-Selector, the training-free rules and the released gates on the same items, and we pair every difference below within them.

Table 17: On a counting task whose evidence fits the budget, KV-Selector is 1.0 point below the attention-mass oracle at 640 tokens, and the registered prediction of a gap of at least 20 points fails. Accuracy (%) over 100 items on Qwen3-8B with every layer restricted, 4,096-token prompts. At 640 tokens every paired interval between KV-Selector, the oracle and the three training-free rules includes zero. At 256 tokens the only such interval that excludes zero is the oracle's 11.0-point lead over the Corner rule. Always giving the most common answer scores 25.0%. KV-Selector takes its feature statistics from its own input. No chosen blocks was not run at 256 tokens.

Configuration	640 tokens	256 tokens
Full attention	64.0	64.0
Attention-mass oracle	60.0	46.0
KV-Selector	59.0	41.0
Mean-key rule	58.0	40.0
Quest	52.0	39.0
Corner rule	52.0	35.0
Random blocks	12.0	15.0
No chosen blocks	17.0	–

The extra feature does not close the coarse-block retrieval gap. In these runs KV-Selector scores 71.3% on RULER at blocks of 64 and 640 tokens, 0.9 points below its score in Table 3, and the variant shows no clear difference from it (paired interval -1.0 to $+1.1$ points). The variant stays 3.2 points below Quest, 6.6 below the released SeerAttention-R gate and 8.0 below the Mean-key rule, all with intervals excluding zero. At 2,048 tokens it matches KV-Selector at 90.4%, shows no clear difference from Quest and the gate, and stays 1.2 points below the Mean-key rule.

The variant helps language modelling slightly. On PG19 at blocks of 64 and 2,048 tokens it is 0.003 nats below KV-Selector, 0.008 below the gate and 0.011 below the Mean-key rule, all with intervals excluding zero, and at 640 tokens it equals KV-Selector. At blocks of 16 it stays within 0.002 nats of KV-Selector on PG19 on all four models, and it is 0.002 lower at both budgets on the 8k and 16k books that its Qwen3-8B run covers. At 640 tokens it moves RULER by -0.3 to $+0.6$ points and LongBench by -0.2 to $+0.7$, with one interval excluding zero on each. It leads the Mean-key rule on RULER by 1.5 to 1.6 points on three models, while on Qwen3-8B the Mean-key rule leads KV-Selector by 1.2 points in these runs, against 0.8 in the main run.

Besides the training windows, which we suspect in Section 4.3, two causes of the deficit remain untested, the training target and the feature statistics fixed from those windows. The statistics alone move RULER by 4.2 points at blocks of 64 (Appendix C.10).

C.12 Two-pass selection

KV-Selector chooses a layer's blocks once, before reading any of them. A two-pass selector would choose again after a first sparse read. Two tests, with predictions and decision rules registered in advance, found no headroom for such a second look, so we did not build one.

The first test gives the scorer one more input per head, the attention mass that the previous layer put on each block when it read only its own chosen blocks, which costs nothing at decoding time. We restricted one layer at 20, 50 or 80% of depth on Qwen3-1.7B and SmolLM2-1.7B, at blocks of 16 and 64, 640 tokens, 4,096-token windows and two seeds. The input changes mean gap closed by $+0.12$ and -0.21 points, and both seeds favour it in only 3 and 2 of 6 cells. Our rule for headroom required more than 1 point, with both seeds agreeing in at least 4 of 6 cells. The previous layer's full attention, which a sparse decoder lacks, gains 1.25 and 2.23 points, mostly at blocks of 64.

The second test asks whether scoring each block on its own misses evidence spread over the whole text. Each item names a target word 1 to 5 times among three distractor words in 4,096 tokens of wikitext-103, and the model answers how often the target appears (Table 17). At 640 tokens every restricted configuration discards at least 84.4% of the context at the answer positions, and random blocks fall below the rate of always giving the most common answer, so the answer depends on which blocks a rule keeps.

Table 18: Bytes read and arithmetic per generated token, summed over layers, at 130,048 tokens with every layer selected, blocks of 16 and a budget of 640 tokens: a count from each model's attention geometry, not a measurement. Values counted at 2 bytes; 1 MB is 2^{20} bytes, as in Table 20; 1 FLOP is half a multiply-add. Selection reads both landmarks of every block and the chosen tokens' keys and values; KV-Selector adds its MLP on every block.

	Full attention		Quest, Corner rule		KV-Selector	
Model	MB read	GFLOP	MB read	GFLOP	MB read	GFLOP
MiniCPM5-2B	5,593	44.7	377	3.0	377	7.3
Granite-4.2-3B	10,654	53.3	718	3.6	718	9.6
K2-Horizon-3.7B	19,176	76.7	1,293	5.2	1,293	10.0

We had predicted that KV-Selector, and the training-free rules with it, would fall at least 20 points behind the oracle. KV-Selector falls 1.0 point behind (interval -6 to $+8$ points), and the training-free rules 2 to 8 points. Both predictions failed, so we ran neither follow-up that depended on them, a second pass within the layer and a fixed counting feature. The counting evidence, at most 14 sentences, fits inside the budget, so this test says nothing about common-word extraction, whose evidence exceeds it.

C.13 Decode latency and offloading

Since every block is scored, one decoding step of one layer reads the budget plus all landmarks. At a budget of 640 tokens and a context of 128k tokens this is 6.7% of the cache at blocks of 16, 2.1% at 64 and 1.3% at 64 with a mean key as the only landmark, and 14.1, 9.4 and 8.6% at 8k tokens. The landmarks take 1.56% of the cache memory at blocks of 64.

Table 18 counts the bytes and arithmetic of one generated token at 130,048 tokens with every layer selected. Quest, the Corner rule and KV-Selector read the same 6.7% of full attention's bytes. KV-Selector's MLP runs on every block and raises the arithmetic from 6.7% of full attention's, for Quest and the Corner rule, to 13.0 to 18.1%, or 8 to 19 FLOP per byte read against 4 to 8 for full attention. These are counts; we have not built a fused kernel, and the measured overhead of our code comes from its many small operations (below).

Table 19 gives every model and length behind the timings of Section 4.5 with the cache in GPU memory. Full attention takes 15.2 to 45.7 ms per token, and selection adds 18.1 to 28.5 ms at every length. We attribute this nearly constant cost to the many small operations per layer (reading landmarks, scoring, choosing the top K, gathering), since it barely changes with the data read. Reading fewer tokens does save time where reading dominates, as the same selection code keeping every token takes 122.7 ms per token at 131,072 tokens on K2-Horizon-3.7B, against 66.9 ms for KV-Selector reading 640.

With the cache in host memory (Table 20), the speed-up over a full copy grows with context length and with the model's cache size, from 10.6 times at 32k tokens to 34.8 times at 128k on K2-Horizon-3.7B. At a budget of 2,048 tokens it is 6.9, 14.5 and 25.0 times at 128k on MiniCPM5-2B, Granite-4.2-3B and K2-Horizon-3.7B.

Scoring on the GPU is what turns the smaller copy into speed. With the landmarks scored on the CPU in host memory, decoding at 128k tokens is 3.7 to 5.9 times slower, since the CPU's scoring time grows with the number of blocks. Every GPU-scored run matched full attention masked to the same tokens to a relative error of at most 3.7×10^{-3}, and the CPU and GPU scorers chose the same blocks. The timed runs select in every layer and standardise each query's features over its own blocks, a statistic no quality run uses; Table 23 gives quality with every layer selected on PG19 at 131k tokens, with prompt and fixed statistics, beside the every-layer study and the strict passkey runs at 4,096 tokens (Appendix C.5), and Table 24 sets that quality beside the cost.

Table 21 gives the peak GPU memory of decoding in the same timing runs. With the cache in host memory, KV-Selector's peak lies 0.3 to 8.4 GB below that of full attention with its cache on the GPU. That is less than the size of the cache, because our implementation allocates temporary

Table 19: With the whole cache in GPU memory, KV-Selector decodes more slowly than full attention at every context length up to 128k tokens. Median milliseconds per generated token on one H200, plain PyTorch without a fused kernel, budget 640 tokens. "Scoring only" computes KV-Selector's choice and discards it, and its excess over full attention, "Overhead", is the cost of the selection machinery.

Model	Context tokens	Full ms/token	Corner ms/token	KV-Selector ms/token	Scoring only ms/token	Overhead ms/token
Qwen3-1.7B	8,192	15.8	30.3	35.4	33.9	+18.1
Qwen3-1.7B	16,384	15.2	28.4	34.3	33.5	+18.2
Qwen3-1.7B	32,768	15.7	32.0	35.4	34.5	+18.8
Qwen3-8B	8,192	20.6	38.6	45.9	44.9	+24.3
Qwen3-8B	16,384	20.8	38.9	46.2	45.2	+24.5
Qwen3-8B	32,768	21.0	43.1	46.1	45.2	+24.2
MiniCPM5-2B	8,192	17.8	38.5	46.6	45.0	+27.2
MiniCPM5-2B	16,384	17.6	36.5	45.6	44.8	+27.2
MiniCPM5-2B	32,768	17.8	38.8	46.7	46.0	+28.3
MiniCPM5-2B	65,536	17.8	39.1	47.6	46.4	+28.5
MiniCPM5-2B	130,048	19.5	43.0	48.3	47.4	+27.9
Granite-4.2-3B	8,192	18.5	38.7	46.6	44.5	+26.0
Granite-4.2-3B	16,384	17.7	38.1	45.7	45.1	+27.4
Granite-4.2-3B	32,768	17.8	39.2	45.7	44.6	+26.9
Granite-4.2-3B	65,536	19.6	43.2	47.2	45.1	+25.5
Granite-4.2-3B	130,048	29.5	52.0	56.8	56.7	+27.2
K2-Horizon-3.7B	8,192	16.5	33.5	41.3	40.3	+23.8
K2-Horizon-3.7B	16,384	16.5	34.1	41.7	40.6	+24.1
K2-Horizon-3.7B	32,768	18.1	38.2	41.5	40.5	+22.4
K2-Horizon-3.7B	65,536	27.3	46.6	50.0	49.1	+21.8
K2-Horizon-3.7B	131,072	45.7	64.8	66.9	66.1	+20.4

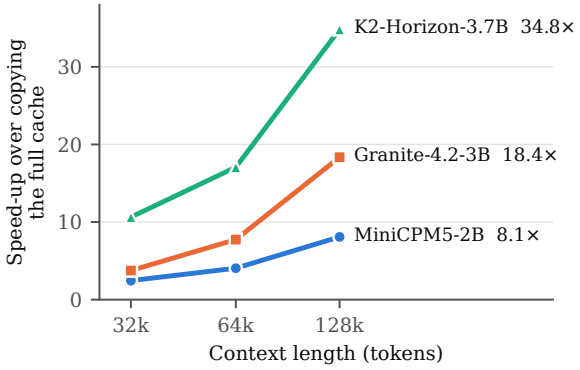

Figure 3: With the KV cache in host memory on one H200, KV-Selector decodes 8.1 to 34.8 times faster at 128k tokens than synchronously copying the whole cache at every step. KV-Selector reads 640 tokens, scores the blocks on the GPU and copies only the chosen ones, 214 times fewer bytes per token at 128k. Table 20 gives every length.

memory during decoding that grows with the context, 7.5 GB at 128k tokens on K2-Horizon-3.7B. Full attention cannot keep its cache on a GPU whose memory lies between the two peaks; there its remaining option is to copy the whole cache at every step, and KV-Selector decodes 2.4 to 34.8 times faster than that copy. The Corner rule reaches the same peaks and is faster. The ranges are read off measured peaks. Decoding under a real memory limit confirms three of them (Table 22): full attention with its cache on the GPU runs out of memory, while KV-Selector fits and decodes 19 to 41 times faster than the whole-cache copy, and the Corner rule again reaches the same peak and

Table 20: Copying only the chosen blocks from host memory decodes 2.4 to 34.8 times faster than copying the whole cache, and the speed-up grows with context length. Budget 640 tokens, KV-Selector scored on the GPU. "Full, copied" copies the whole cache to the GPU at every step, and "Full, on GPU" keeps it in GPU memory. The last column gives megabytes copied per token by KV-Selector and by the full copy.

Model	Context tokens	Full, on GPU ms/token	Full, copied ms/token	KV-Selector ms/token	Speed-up vs copied	Bytes copied MB/token
MiniCPM5-2B	32,768	17.5	155.6	63.6	2.4×	24.9 / 1,344
MiniCPM5-2B	65,536	17.5	260.8	64.4	4.0×	24.9 / 2,688
MiniCPM5-2B	130,048	19.2	523.4	64.8	8.1×	24.9 / 5,334
Granite-4.2-3B	32,768	17.2	242.4	64.7	3.7×	47.5 / 2,560
Granite-4.2-3B	65,536	19.3	500.3	64.6	7.7×	47.5 / 5,120
Granite-4.2-3B	130,048	29.6	1209.1	65.9	18.4×	47.5 / 10,160
K2-Horizon-3.7B	32,768	17.8	683.8	64.3	10.6×	85.5 / 4,608
K2-Horizon-3.7B	65,536	27.0	1105.0	64.8	17.1×	85.5 / 9,216
K2-Horizon-3.7B	130,048	45.2	2290.8	65.8	34.8×	85.5 / 18,288

Table 21: With the cache in host memory, KV-Selector needs less GPU memory than full attention with its cache on the GPU, and on GPUs in between it is the fast way to decode. Peak GPU memory while decoding on one H200, budget 640 tokens. "GPU memory where only offloading fits" is the range in which full attention cannot keep its cache on the GPU but KV-Selector with the cache in host memory fits; the last column is KV-Selector's speed-up there over copying the whole cache at every step. Read off measured peaks; Table 22 decodes three settings under a real limit.

			Peak GPU memory (GB)				
Model	Context	KV cache (GB)	Cache on GPU	KV-Selector, cache in host	Copy whole cache	GPU memory where only offloading fits (GB)	KV-Selector vs copy
Qwen3-1.7B	32,768	3.5	6.8	4.8	3.9	4.8–6.8	10.2×
MiniCPM5-2B	32,768	1.3	6.0	5.7	4.9	5.7–6.0	2.4×
MiniCPM5-2B	65,536	2.6	7.4	6.7	5.2	6.7–7.4	4.0×
MiniCPM5-2B	130,048	5.2	10.0	8.7	5.6	8.7–10.0	8.1×
Granite-4.2-3B	32,768	2.5	9.4	8.3	7.3	8.3–9.4	3.7×
Granite-4.2-3B	65,536	5.0	11.9	9.7	7.7	9.7–11.9	7.7×
Granite-4.2-3B	130,048	9.9	16.9	12.6	8.6	12.6–16.9	18.4×
K2-Horizon-3.7B	32,768	4.5	14.0	11.9	10.3	11.9–14.0	10.6×
K2-Horizon-3.7B	65,536	9.0	18.6	14.3	11.1	14.3–18.6	17.1×
K2-Horizon-3.7B	130,048	17.9	27.6	19.2	12.7	19.2–27.6	34.8×

is faster. The limit applies to decoding only: the prefill, which our code runs with the cache on the GPU, would need its own offloading to fit.

Table 23 measures quality at the length of the timing runs: PG19 NLL on 10 books of 131,072 tokens, a 129,024-token prompt followed by 2,048 scored tokens, both with the first two layers dense, as in the benchmarks, and with selection in every layer, as in the timing runs. On all three models, KV-Selector with fixed statistics has the lowest NLL of the compared rules at both budgets and in both configurations. Its lead over the strongest training-free rule is 0.018 to 0.027 nats at 640 tokens on MiniCPM5-2B and Granite-4.2-3B and 0.070 to 0.076 on K2-Horizon-3.7B, and 0.006 to 0.047 nats at 2,048 tokens. Its paired interval over books excludes zero in 11 of the 12 rows, all but MiniCPM5-2B at 2,048 tokens with two dense layers.

Table 24 sets quality with every layer selected beside its cost, from a further timing run that also times KV-Selector with fixed statistics, its benchmark configuration. With the cache in host memory and fixed statistics, KV-Selector decodes at 62.1 to 64.2 ms per token with a peak of 8.7 to 19.2 GB of GPU memory, 2.0 to 2.5 ms faster than with per-query statistics, whose mean and deviation over the blocks' features it replaces by constants. The Corner rule reaches the same peak 1.7 to 7.3 ms per token faster, with a PG19 NLL 0.049 to 0.149 nats above that of KV-Selector with fixed statistics. Full attention needs 19.3 to 45.4 ms per token and 10.0 to 27.6 GB with its cache on the GPU, or 517 to 2,597 ms per token when it copies the whole cache from host memory, at an NLL 0.11 to 0.13 nats below KV-Selector with fixed statistics. In this run, per-query KV-Selector and the Corner rule stay within 1.0 ms per token of their times in Table 20, and the whole-cache copy within 13%.

Table 22: Under a real GPU memory limit, full attention with its cache on the GPU runs out of memory, while KV-Selector with the cache in host memory fits and decodes 19 to 41 times faster than copying the whole cache. The limit applies after the prefill. One H200, budget 640 tokens; KV-Selector's peak GPU memory in parentheses.

Model	Context tokens	Limit (GB)	Full attention, cache on GPU	Copy whole cache ms/token	KV-Selector ms/token (peak GB)	Corner rule ms/token	KV-Selector vs copy
Granite-4.2-3B	130,048	16	out of memory	1,500	69.0 (12.6)	61.2	21.8×
K2-Horizon-3.7B	130,048	24	out of memory	2,684	65.6 (19.2)	61.5	40.9×
K2-Horizon-3.7B	65,536	16	out of memory	1,275	67.4 (14.3)	58.3	18.9×

Table 23: At 131k tokens on the three timed models, KV-Selector keeps the lowest PG19 NLL of the compared rules, with the first two layers dense as in the benchmarks and with selection in every layer as in the timing runs. NLL in nats over the last 2,048 tokens of 10 books (lower is better); bold marks the lowest NLL other than full attention. "Lead" is KV-Selector (fixed statistics) minus the strongest training-free rule, paired over books; * marks a 95% interval that excludes zero.

Model	Dense layers	Budget	Full	KV-Selector	KV-Selector, prompt stat.	Mean-key	Corner	Quest	Lead	95% interval
MiniCPM5-2B	2 (benchmarks)	640	2.812	**2.933**	2.949	2.960	2.973	3.130	-0.027*	[-0.057, -0.005]
MiniCPM5-2B	2 (benchmarks)	2,048	2.812	**2.881**	2.890	2.887	2.890	2.948	-0.006	[-0.017, +0.004]
MiniCPM5-2B	0 (timing runs)	640	2.811	**2.926**	2.944	2.949	2.982	3.167	-0.023*	[-0.048, -0.003]
MiniCPM5-2B	0 (timing runs)	2,048	2.811	**2.879**	2.896	2.889	2.894	2.963	-0.009*	[-0.019, -0.002]
Granite-4.2-3B	2 (benchmarks)	640	2.506	**2.615**	2.637	2.634	2.667	2.797	-0.019*	[-0.028, -0.011]
Granite-4.2-3B	2 (benchmarks)	2,048	2.506	**2.568**	2.578	2.578	2.584	2.640	-0.009*	[-0.017, -0.001]
Granite-4.2-3B	0 (timing runs)	640	2.506	**2.620**	2.633	2.637	2.668	2.799	-0.018*	[-0.026, -0.011]
Granite-4.2-3B	0 (timing runs)	2,048	2.506	**2.572**	2.578	2.583	2.587	2.644	-0.011*	[-0.017, -0.004]
K2-Horizon-3.7B	2 (benchmarks)	640	2.609	**2.746**	2.765	2.816	2.841	3.091	-0.070*	[-0.137, -0.031]
K2-Horizon-3.7B	2 (benchmarks)	2,048	2.609	**2.670**	2.679	2.715	2.697	2.781	-0.027*	[-0.036, -0.014]
K2-Horizon-3.7B	0 (timing runs)	640	2.610	**2.743**	2.769	2.819	2.892	3.150	-0.076*	[-0.110, -0.053]
K2-Horizon-3.7B	0 (timing runs)	2,048	2.610	**2.669**	2.684	2.726	2.716	2.793	-0.047*	[-0.058, -0.035]

C.14 Measurement validity

Correctness checks. The float64 gates of Appendix A passed with residuals far below their tolerance of 10^{-4}. Chunked teacher forcing matched token-by-token decoding to at most 1.4×10^{-6} in NLL per configuration, except per-head KV-Selector on Qwen3-8B at 3.0×10^{-5}. We traced that residual to float32 arithmetic in the model's normalisation layers, which differs in the last bit between chunked and single-query runs and flips one near-tied block choice in one head. Generation from a shared prompt cache reproduced every greedy sequence of a fresh cache for all 16 configurations on all four models, with forced-token logits within 1.3×10^{-5}.

Kernel noise floor. With one layer restricted, the mask that keeps every block gives a KL of 4×10^{-4} to 8×10^{-4} on Qwen3-1.7B, SmolLM2-1.7B and Gemma-3-1B, 33 to 70% of the oracle's KL on the first two and 2 to 11% on Gemma-3-1B. With every full-attention layer of the last six models restricted, the floor is 6×10^{-4} to 4.3×10^{-3}, 1 to 14% of the oracle's KL. The floor adds a nearly constant offset to every configuration's KL, which cancels in Equation 4. Against the keep-every-block run, gap closed moves by at most 2.0 points in any one-layer cell and 0.9 in any every-layer cell, and the best rule changes in 1 of the 48 one-layer cells of the last six models.

Attention-mass oracle. With every layer restricted, the oracle still picks each layer's blocks from the unrestricted model's attention, one layer at a time, and other configurations can beat it. In the every-layer study its wikitext-103 loss falls below the unrestricted model's by 0.018 and 0.016 nats on Qwen3.8-27B at blocks of 16 and 64, and by 0.029 and 0.036 on Granite-4.2-3B. On PG19 at 640 tokens it has a lower NLL than KV-Selector only on the two Qwen3 models, and KV-Selector is 0.011 nats below it on K2-Horizon-3.7B. With 64k-token prompts on MiniCPM5-2B at 640 tokens, the oracle is 0.019 nats above KV-Selector (interval -0.036 to $+0.068$). On RULER it stays 4.2 to 7.1 points above KV-Selector, so retrieval headroom remains. We call it the attention-mass oracle and not an upper bound.

Table 24: Quality and cost with every layer selected at 128k tokens, reading 640 tokens per query. Milliseconds per generated token and peak GPU memory come from one timing run on one H200 at a 130,048-token context that times every row except prompt statistics and the Mean-key rule; PG19 NLL in nats from the 10 books of 131,072 tokens in Table 23. The three statistics differ only in the mean and deviation used to standardise the features. "–": not measured.

Configuration	Cache	ms/token	Peak GPU (GB)	PG19 NLL
MiniCPM5-2B				
Full attention	GPU	19.3	10.0	2.811
Full attention	host, whole cache copied	516.8	5.6	2.811
KV-Selector, per-query statistics	host, chosen blocks copied	64.6	8.7	–
KV-Selector, prompt statistics	host, chosen blocks copied	–	–	2.944
KV-Selector, training statistics	host, chosen blocks copied	62.1	8.7	2.926
Corner rule	host, chosen blocks copied	54.8	8.7	2.982
Mean-key rule	host, chosen blocks copied	–	–	2.949
Granite-4.2-3B				
Full attention	GPU	29.4	16.9	2.506
Full attention	host, whole cache copied	1180.2	8.6	2.506
KV-Selector, per-query statistics	host, chosen blocks copied	66.7	12.6	–
KV-Selector, prompt statistics	host, chosen blocks copied	–	–	2.633
KV-Selector, training statistics	host, chosen blocks copied	64.2	12.6	2.620
Corner rule	host, chosen blocks copied	59.2	12.6	2.668
Mean-key rule	host, chosen blocks copied	–	–	2.637
K2-Horizon-3.7B				
Full attention	GPU	45.4	27.6	2.610
Full attention	host, whole cache copied	2597.0	12.7	2.610
KV-Selector, per-query statistics	host, chosen blocks copied	64.8	19.2	–
KV-Selector, prompt statistics	host, chosen blocks copied	–	–	2.769
KV-Selector, training statistics	host, chosen blocks copied	62.8	19.2	2.743
Corner rule	host, chosen blocks copied	61.2	19.2	2.892
Mean-key rule	host, chosen blocks copied	–	–	2.819