LINEAR INDEXER: UNDERSTANDING AND ACCELERATING DEEPSEEK SPARSE ATTENTION

Anonymous authors
Paper under double-blind review

ABSTRACT

DeepSeek's Lightning Indexer scores every query against all visible keys, incurring quadratic prefill cost. In our DeepSeek-V4-Flash profile at 256K context, it accounts for 33.3% of prefill GPU busy time. Surprisingly, we find that removing its head-wise ReLUs largely preserves retrieval. This observation yields *Linear Indexer*, a training-free method that folds the 64 weighted query heads exactly into one before the global scan, reducing 64 query–key dot products per key to one. The resulting linear score directly selects keys for attention; for higher retrieval recall, the original scorer can rerank an enlarged linear shortlist. Our analysis links the high recall to learned coupling between stable query–key signs and head weights, a pattern observed across domains, context lengths, checkpoints, and Indexer architectures. On V4-Flash, Linear Indexer maintains performance comparable to the original model on LongBench-v2 and RULER at context lengths up to 256K. At 256K on NVIDIA H20 GPUs, direct mode achieves a $10.4\times$ indexer speedup and a $1.39\times$ speedup in end-to-end prefill time to first token (TTFT).

1 INTRODUCTION

Self-attention (Vaswani et al., 2017) is a core component of modern Transformers, but its quadratic cost makes long-context inference expensive. Sparse attention alleviates this bottleneck by attending only to a small subset of important tokens. DeepSeek Sparse Attention (DSA) (DeepSeek-AI, 2025) uses a learned *Lightning Indexer*, a lightweight 64-head ReLU scorer that ranks candidate tokens and selects the Top-K. However, because the Indexer still scans the full visible context for every query, its cost grows with context length and becomes a substantial cost during long-context prefill.

Existing efforts reduce this cost by shrinking Indexer computation along architectural axes. In frontier models, DeepSeek-V4 (DeepSeek-AI, 2026) introduces Compressed Sparse Attention (CSA), which uses roughly $4\times$ fewer indexed keys, while GLM-5 (GLM, 2026) and Hy4-preview (Tencent Hy Team, 2026) share one Indexer across four layers via IndexCache reuse (Bai et al., 2026). Recent methods explore complementary axes: HISA (Xu et al., 2026) hierarchically prunes the key space, MISA (Zhou et al., 2026) routes only 8 of the 64 Indexer heads, PIVOT (Liu et al., 2026) reduces redundant computation across queries, and LongCat Sparse Attention (Zan et al., 2026) combines reductions along multiple dimensions. DeepSeek-V4.1 (DeepSeek-AI, 2026) further introduces CSA2, combining cross-layer key and index reuse with hierarchical candidate restriction. Despite their different mechanisms, these approaches reduce indexer computation along existing key, head, query, or layer dimensions, but leave the algebraic form of the original head-wise nonlinear scorer unchanged. This leaves potential redundancy within the scoring function itself unexplored. Surprisingly, we find that removing the head-wise ReLU largely preserves retrieval.

Figure 1 illustrates the central observation. On a six-prompt 64K pilot, the linear score directly recovers 77.1% of the original Top-512 selections, and increasing the proposal budget to $B = 2048$ raises the recall to 98.9%. This observation matters because the indexer score itself is not consumed by the main attention operator; only the induced retrieval decision matters.

As shown in Fig 1(b), removing the head-wise ReLUs makes the score linear in the key. The weighted query heads can be reassociated exactly: $\sum_{h=1}^{64} w_{t,h}(q_{t,h}^\top k_s) = (\sum_{h=1}^{64} w_{t,h} q_{t,h})^\top k_s$. Thus, the weighted query heads can be combined before the global key scan, reducing the scan from 64 head-wise query–key dot products per candidate to a single dot product.

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.

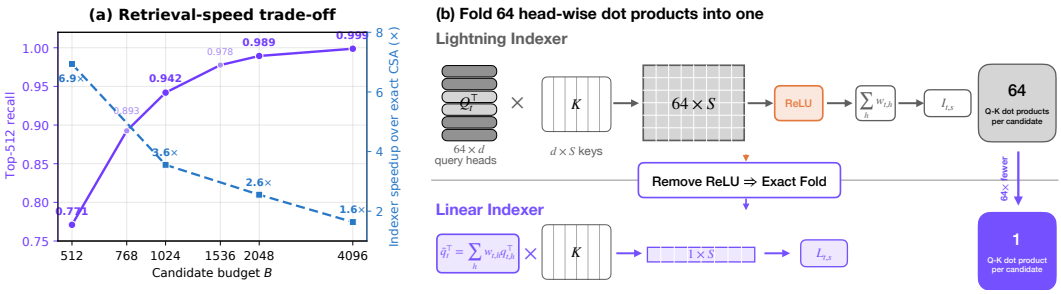


Figure 1: **The Linear Indexer preserves retrieval while accelerating the global scan. (a)** On a six-prompt 64K pilot, increasing B from 512 to 2048 raises Top-512 recall from 77.1% to 98.9%; at 64K on one H20, the indexer speedups over exact CSA are $6.9\times$ and $2.6\times$. **(b)** Removing the head-wise ReLUs makes the score linear in the key, allowing the 64 weighted query heads to be folded before the global scan and reducing 64 query–key dot products per candidate to one.

We exploit this retrieval redundancy with the **training-free** *Linear Indexer*. In its simplest form, it directly returns the linear Top-K ($B = K$). For higher retrieval recall, it can instead retrieve $B > K$ candidates and apply the original 64-head ReLU scorer only to this shortlist before final Top-K selection. Fig. 1(a) also shows that, at 64K context, the $B = 512$ and $B = 2048$ settings achieve $6.9\times$ and $2.6\times$ Indexer speedups over exact CSA, respectively.

Why does such a simple linear score preserve retrieval so well? A symmetric toy model provides a partial explanation: even without learned structure, the ReLU score naturally favors its signed linear component. Beyond this toy model, many heads in trained Indexers exhibit sign-stable query–key products, while their learned weights tend to have the opposite sign. Removing ReLU introduces score errors mainly through predominantly negative heads, but these error terms tend to rank keys similarly to the positive-head contributions rather than act as random ranking noise. Randomizing the head–weight pairing largely destroys this behavior. These patterns persist across domains, context lengths, and checkpoints, and span token-level DSA, compressed CSA, and hierarchical CSA2, despite differences in layer/head counts, Top-K budgets, key compression, and cross-layer reuse.

Beyond its inference benefits, our findings also motivate training future sparse-attention indexers directly with the folded linear score. The same factorization reduces query–key scoring arithmetic in both forward and backward passes, offering a route to lower indexer training cost on long sequences. We focus here on training-free inference acceleration and leave native training to future work.

Contributions. (i) We uncover strong *linear retrieval redundancy* in trained Lightning Indexers: removing the head-wise ReLUs preserves much of the original Top-K set and enables high-recall linear proposals without retraining. (ii) We explain this behavior through a symmetric toy model and learned head–weight structure, which persists across domains, context lengths, checkpoints, and three distinct Indexer designs: token-level DSA, compressed CSA, and hierarchical CSA2. (iii) We introduce the training-free *Linear Indexer*, with direct and refined retrieval modes, achieving a stronger retrieval–compute tradeoff, no significant downstream degradation through 256K, and speedups of up to $10.4\times$ for the indexer and $1.39\times$ for prefill time to first token.

2 Preliminaries

DeepSeek sparse attention. DeepSeek-V3.2 (DeepSeek-AI, 2025) applies the Lightning Indexer to token-level key–value entries and selects $K = 2048$ tokens. DeepSeek-V4-Flash (DeepSeek-AI, 2026) instead uses CSA, which forms one indexed entry for every four tokens and selects $K = 512$ entries, alongside separate sliding window and sink paths. DeepSeek-V4.1-Flash (DeepSeek-AI, 2026) further introduces CSA2, which uses a 32-head Top-512 Indexer and combines cross-layer KV and index reuse with hierarchical candidate restriction for later reindexing.

Lightning Indexer. For query t, candidate s, and head h, let $q_{t,h}, k_s \in \mathbb{R}^d$ and let $w_{t,h}$ be a query-dependent head weight, with $H = 64$ and $d = 128$ in DSA and CSA. The Lightning Indexer scores and selects candidates as

$$I_{t,s} = \sum_{h=1}^{H} w_{t,h} \, \mathrm{ReLU}(q_{t,h}^\top k_s), \qquad \mathcal{T}_t = \mathrm{TopK}_s(I_{t,s}) \, . \tag{1}$$

Let S denote the number of indexed keys visible to a query. The Lightning Indexer costs $O(SHd)$ per query and $O(S^2Hd)$ over prefill, and therefore remains quadratic in context length.

3 Why Does Linear Retrieval Work?

We now ask why removing the head-wise ReLUs preserves retrieval so well. Our analysis separates two sources of this behavior. First, a symmetric toy model shows that the ReLU score has an intrinsic preference for its signed linear component, providing a partial explanation even without learned structure. Beyond this intrinsic bias, trained Indexers exhibit additional learned structure: compressed keys form a concentrated cone that induces sign-stable heads, while learned head weights tend to take the opposite sign. The negative-head terms introduced by removing ReLU then tend to rank keys similarly to the positive-head contributions rather than act as random noise. Targeted interventions that disrupt only the learned head–weight pairing sharply reduce linear retrieval recall.

3.1 A Symmetric Toy Model Favors the Linear Component

Using $\mathrm{ReLU}(x) = (x + |x|)/2$, the exact indexer score can be decomposed as

$$I_{t,s} = \tfrac{1}{2}\Bigg(\underbrace{\sum_h w_{t,h} q_{t,h}^\top k_s}_{L_{t,s}} + \underbrace{\sum_h w_{t,h} |q_{t,h}^\top k_s|}_{A_{t,s}} \Bigg). \tag{2}$$

where L is the signed linear component and A is the nonlinear absolute-value component.

Under the symmetric assumptions of Proposition B.1, the exact score is more strongly correlated with L than with A:

$$\mathrm{Corr}(I, A) < 1/\sqrt{2} < \mathrm{Corr}(I, L). \tag{3}$$

For Gaussian activations, the correlation $\mathrm{Corr}(I, L)$ becomes $\sqrt{\pi/[2(\pi-1)]} \approx 0.856$. The complete statement and proof are deferred to Proposition B.1. This toy model shows that part of the observed linear preference can arise from the ReLU decomposition itself, but does not explain the high Top-K retrieval recall of the trained Indexer.

3.2 Sign Structure Matters for Linear Retrieval

We first examine whether this linear preference also holds in the trained Indexer, where the symmetric assumptions no longer hold. We use the same six-prompt 64K DeepSeek-V4-Flash-0731 pilot introduced in Fig. 1; complete capture and sampling protocols are provided in Appendix C.

Despite substantial violations of the assumptions of the toy model, the signed linear score retains strong ranking signal across all 21 CSA layers. As shown in Fig. 2, its mean Pearson correlation with the exact scores is 0.922; it outperforms the absolute-value score in correlation in every layer and in Top-512 recall in 19 of 21 layers. By contrast, **the fully unsigned score has a mean correlation of -0.205 and substantially lower recall**, showing that learned signs carry important retrieval structure. We next characterize this sign structure and its interaction with the learned head weights.

3.3 A cone-shaped key manifold induces sign-stable heads

Since ReLU acts directly on the query–key products $q_{t,h}^\top k_s$, we begin by examining their signs. In the trained Indexer, many heads exhibit an almost fixed activation sign over the empirical key distribution.

For each layer–head pair, we measure the fraction of observed query–key products that are positive:

$$p_h^{\mathrm{all}} = \Pr_{t,s}\left(q_{t,h}^\top k_s > 0\right).$$

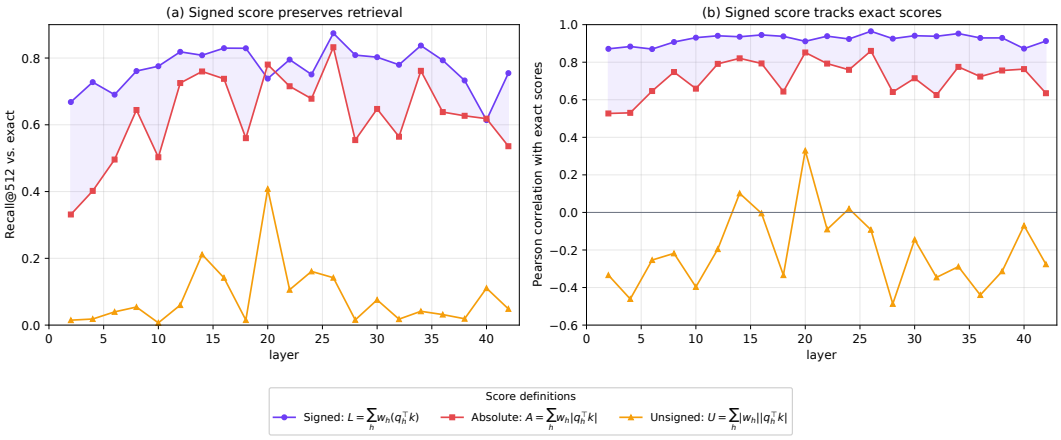


Figure 2: Layer-wise comparison of the signed, absolute-value, and fully unsigned scores in the trained V4 Indexer. **(a)** The signed score achieves higher Top-512 recall than the absolute-value score in 19 of 21 layers and than the fully unsigned score in every layer. **(b)** The signed score exhibits stronger Pearson correlation with the exact scores than both alternatives in every layer. Shading highlights the gap between the signed and absolute-value scores.

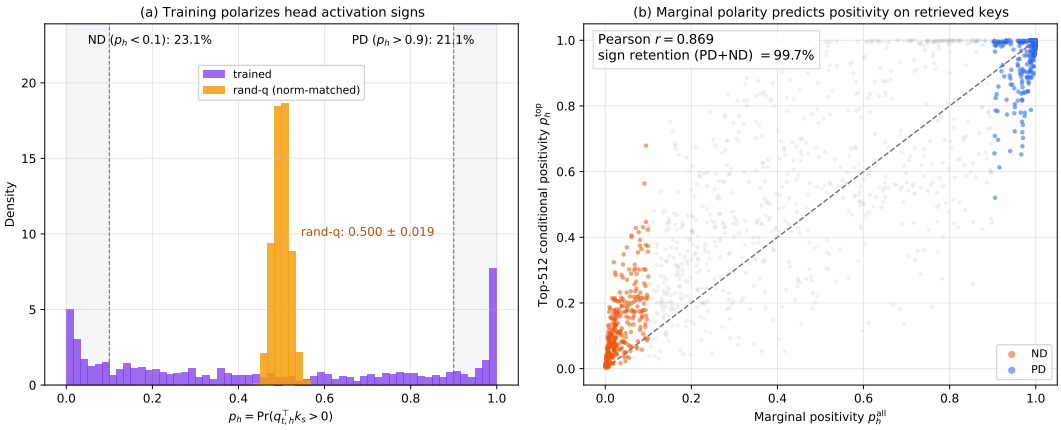


Figure 3: The trained Indexer develops stable signs in its pre-ReLU query–key products. **(a)** For many trained heads, the query–key products are almost always positive or almost always negative, unlike norm-matched random queries. **(b)** The same dominant sign is largely preserved among the exact Top-512 retrieved keys.

We classify a head as negative-dominant (ND) if $p_h^{\text{all}} < 0.1$, positive-dominant (PD) if $p_h^{\text{all}} > 0.9$, and mixed-sign (MID) otherwise. **Nearly half of trained heads have an almost fixed query–key product sign.** As shown in Fig. 3(a), in the trained indexer, 23.1% of heads are ND and 21.1% are PD, producing a clear U-shaped distribution. By contrast, norm-matched random query directions concentrate tightly around 0.5. Head-wise sign stability is therefore a learned property rather than a generic consequence of high-dimensional dot products.

Fig 3(b) shows these dominant signs persist in the retrieval Top-512 entries. Define

$$p_h^{\text{top}} = \Pr_{t,s}\left(q_{t,h}^\top k_s > 0 \,\middle|\, s \in \text{Top-512}(I_t)\right).$$

The sign pattern over all keys strongly predicts that within the exact Top-512 set ($r = 0.869$). Among the 594 PD/ND layer–head pairs, 592 retain the same dominant sign:

$$\Pr\left[\left(p_h^{\text{all}} - \tfrac{1}{2}\right)\left(p_h^{\text{top}} - \tfrac{1}{2}\right) > 0 \,\middle|\, h \in \text{PD} \cup \text{ND}\right] = 592/594 = 99.7\%.$$

All PD heads retain positive dominance, while 99.4% of ND heads retain negative dominance.

To understand where this stable sign behavior comes from, we examine the geometry of the compressed keys. For each layer–prompt cache, let $\mu_k = \mathbb{E}_s[k_s]$ denote its mean compressed-key vector.

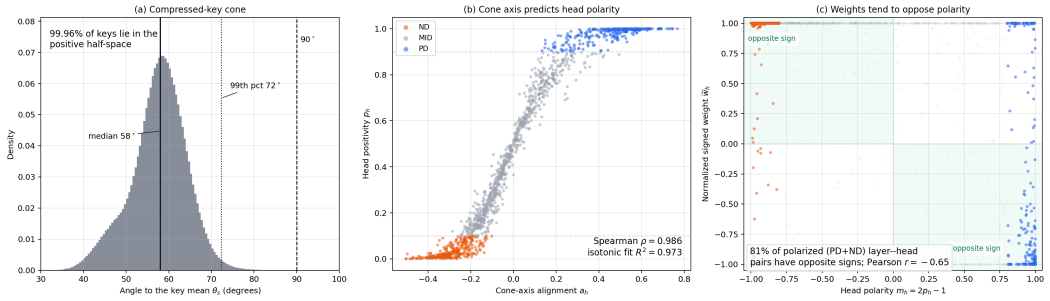


Figure 4: Key geometry and learned head–weight pairing support linear retrieval. **(a)** Compressed keys concentrate around a common mean direction. **(b)** Alignment with this direction predicts the dominant head sign. **(c)** Learned weight signs tend to oppose this sign.

As shown in Fig. 4(a), across 2.06 million compressed keys, 99.96% satisfy $k_s^\top \mu_k > 0$. The median and 99th-percentile angles to μ_k are $58°$ and $72°$, respectively. Thus, across layers and prompts, **compressed keys consistently form a cone lying almost entirely within the positive half-space** defined by their specific mean axis. This geometry makes a head's sign behavior predictable from its position relative to the cone axis. For each layer–head pair, define its mean cone-axis alignment as

$$a_h = \mathbb{E}_t \left[\cos(q_{t,h}, \mu_k)\right].$$

Alignment a_h almost completely orders the observed positivity p_h^{all}: their Spearman correlation is $\rho = 0.986$ (Fig. 4(b)). **Heads pointing into the cone become PD, heads pointing away from it become ND**, and only heads near the cone equator remain MID. The observed sign stability is therefore a direct consequence of the learned query directions interacting with the key cone manifold.

3.4 Learned weights align PD and ND heads into a common direction

Sign stability alone does not explain why the linear score preserves retrieval. PD and ND heads tend to point in opposite directions relative to the key mean, so their contributions could cancel when the weighted queries are summed. We find that the learned head weights systematically take the opposite sign to the dominant head sign, bringing these weighted query directions into alignment.

To quantify this relation, define

$$m_h = 2p_h^{\text{all}} - 1, \qquad \widetilde{w}_h = \frac{\mathbb{E}_t[w_{t,h}]}{\mathbb{E}_t[|w_{t,h}|]}.$$

Here, $m_h \in [-1, 1]$ summarizes the dominant query–key product sign, $m_h \approx +1$ means that the products $q_{t,h}^\top k_s$ are almost always positive, indicating a PD head, whereas $m_h \approx -1$ means they are almost always negative. Likewise, $\widetilde{w}_h \in [-1, 1]$ summarizes the dominant sign of the learned head weight, $\widetilde{w}_h \approx +1$ or -1 means that the weight is predominantly positive or negative, respectively.

Across PD/ND heads, m_h and $\widetilde{w}_h$ have Pearson correlation $r = -0.65$, and **81% of heads have opposite signs** (Fig. 4(c)). The same tendency holds at individual query rows: $w_{t,h}$ and $q_{t,h}^\top \mu_k$ have opposite signs for 80.0% of PD/ND query–head instances, compared with 70.9% across all heads.

Empirically, the query projection onto the mean-key direction and the learned head weight typically have opposite signs:

$$\begin{aligned} &q_h^\top \mu_k > 0, \quad w_h < 0 \qquad \text{for PD heads,} \\ &q_h^\top \mu_k < 0, \quad w_h > 0 \qquad \text{for ND heads,} \end{aligned} \qquad \Longrightarrow \qquad (w_h q_h)^\top \mu_k < 0 \quad \text{in both cases.}$$

Thus, PD and ND heads point in opposite directions before weighting, but their weighted query vectors tend to align. This learned head–weight pairing makes the exactly folded query $\bar{q}_t = \sum_h w_{t,h} q_{t,h}$ a coherent retrieval direction. Its importance is confirmed by direct interventions: **permuting the head weights or randomizing their signs while keeping the trained q and k fixed sharply reduces linear retrieval recall across nearly all layers** (see Appendix D.1).

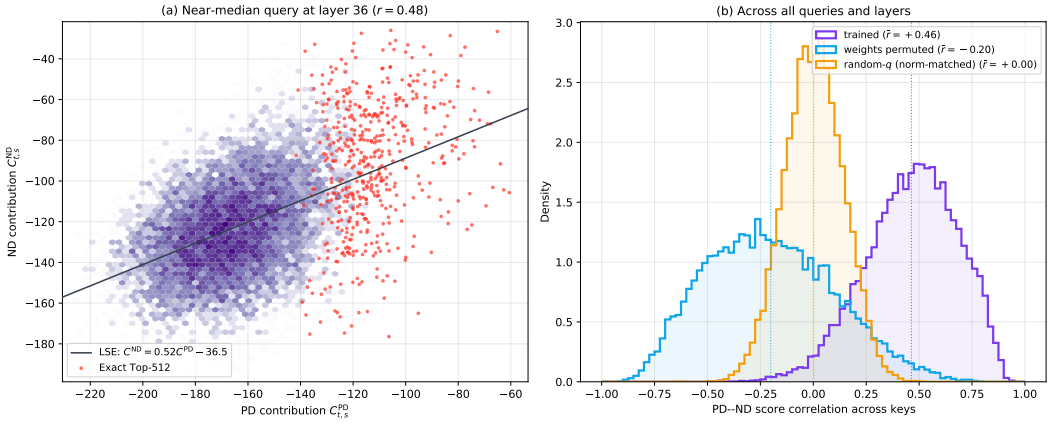


Figure 5: Learned head–weight pairing aligns PD and ND contributions. **(a)** Their contributions are positively correlated; exact Top-512 keys are in red. **(b)** Pairing yields positive PD–ND correlation, whereas weight permutation reverses it and norm-matched random queries yield no correlation.

3.5 Errors introduced by removing ReLU are structured

The preceding analysis explains why the weighted PD and ND head directions become aligned, but a remaining question is whether the score errors introduced by removing ReLU disrupt retrieval. For PD heads, ReLU keeps most query–key products, whereas for ND heads it suppresses most of them. Thus, removing ReLU primarily restores ND-head contributions that are absent from the exact score. The key question is whether these additional terms preserve or arbitrarily perturb the key ranking.

For each query and $G \in \{\mathrm{PD}, \mathrm{ND}\}$, define the contributions of group G as follows:

$$v_t^G := \sum_{h \in G} w_{t,h} q_{t,h}, \qquad C_{t,s}^G := (v_t^G)^\top k_s = \sum_{h \in G} w_{t,h} q_{t,h}^\top k_s.$$

Under this notation, $C_{t,s}^{\mathrm{PD}}$ approximates the PD-head contribution to both the exact and linear scores, whereas $C_{t,s}^{\mathrm{ND}}$ is largely suppressed in the exact score but restored in the linear score. If the restored ND term were arbitrary noise, it would exhibit little agreement with $C_{t,s}^{\mathrm{PD}}$ across keys and would consequently perturb the key ranking.

Instead, **the ND and PD contributions are positively correlated and tend to favor the same high-scoring keys.** Figure 5(a) shows a representative query–layer instance, where the PD and ND contributions have Pearson correlation $r = 0.48$ across all valid keys, with the exact Top-512 in red. Across all 16,128 query–layer instances, the mean per-query Pearson score correlation is $\bar{r} = 0.46$ (Fig. 5(b)), while the mean Spearman rank correlation is $\bar{\rho} = 0.44$ (Fig. 9(a)). Their Top-512 sets overlap by 28%, compared with about 4.1% expected from random Top-512 sets (Fig. 9(b)).

Permuting weights across heads makes both mean correlations negative ($\bar{r} = -0.20$, $\bar{\rho} = -0.19$) and reduces the Top-512 overlap to chance level; norm-matched random queries likewise yield near-zero correlations and chance-level overlap. Complete metric definitions and the rank-correlation and Top-512-overlap distributions are provided in Appendix D.2.

Together, these results show that the score errors introduced by removing ReLU are highly structured rather than random. The key geometry creates stable PD/ND signs, learned head weights align their weighted query directions, and the additional ND terms tend to favor the same high-scoring keys as the PD contributions. This joint key–weight structure explains why the linear score preserves the retrieval ordering of the original nonlinear Indexer.

Generality of the learned structure. The learned structure is not specific to the six-prompt 64K V4 pilot. It holds across domains and context lengths. It also remains stable through training, appearing in the Base checkpoint and persisting after post-training. The same structure recurs throughout DSA Indexers of DeepSeek-V3.2 (DeepSeek-AI, 2025) and CSA2 Indexers of DeepSeek-V4.1-Flash (DeepSeek-AI, 2026), covering both full-scan and hierarchical reindex layers. Detailed results by domain, context length, checkpoint, and model are provided in Appendix E.

4 METHOD: THE LINEAR INDEXER

Folded linear score. The analysis above establishes the signed linear score $L_{t,s}$ as a strong retrieval proxy for the original Lightning Indexer. Unlike the original score $I_{t,s}$, $L_{t,s}$ is linear in the key:

$$L_{t,s} = \sum_{h=1}^{H} w_{t,h} q_{t,h}^\top k_s = (\sum_{h=1}^{H} w_{t,h} q_{t,h})^\top k_s = \bar{q}_t^\top k_s, \qquad \bar{q}_t = \sum_{h=1}^{H} w_{t,h} q_{t,h}. \tag{4}$$

The H weighted query heads are folded before the global key scan, replacing H head-wise dot products per key with one. The transformation can apply to DSA, CSA and CSA2 without training.

Direct and refined retrieval. For each query t and a candidate budget $B \geq K$, define $\mathcal{C}_B(t) = \mathrm{TopB}_s(L_{t,s})$. The Linear Indexer returns

$$\mathcal{T}_K^{\mathrm{LI}}(t) = \begin{cases} \mathcal{C}_K(t), & B = K, \\ \mathrm{TopK}_{s \in \mathcal{C}_B(t)}(I_{t,s}), & B > K. \end{cases} \tag{5}$$

$B = K$ gives *direct mode*, whereas $B > K$ invokes *refined mode* by reranking $\mathcal{C}_B(t)$ with the original scorer. The proposal budget B is the parameter that controls the retrieval–compute tradeoff. The complete procedure is summarized in Algorithm 1 in Appendix F.

Computational cost. For a single query over S indexed keys, the original Lightning Indexer requires $O(SHd)$ arithmetic, excluding selection overhead. The Linear Indexer folds the weighted query heads in $O(Hd)$ and scans all keys once in $O(Sd)$. In refined mode, nonlinear rescoring over the proposal set adds $O(BHd)$, reducing $O(SHd)$ to $O(Hd + Sd + BHd)$. The corresponding ideal arithmetic reduction is $\frac{SH}{H+S+BH} = \frac{H}{1+H/S+BH/S}$, which approaches H as S grows for fixed B. In direct mode, refinement vanishes and the cost reduces to $O(Hd + Sd)$.

5 EXPERIMENTS

5.1 SETUP

Our primary model is DeepSeek-V4-Flash-0731, which contains 21 CSA layers, 64 indexer heads with dimension $d = 128$, a compression ratio of 4, and a final retrieval budget $K = 512$. Cross-model experiments use DeepSeek-V3.2, whose 61 DSA indexer layers operate on uncompressed tokens with $K = 2048$. On V4, candidate-level analyses use $B \in \{512, 1024, 2048, 4096\}$, while served TTFT and downstream evaluations use $B \in \{512, 1024, 2048\}$. Candidate-level analyses are performed on matched V4 and V3.2 capture suites; experiment-specific context lengths, query sampling, and capture protocols are described in Appendix C.

Indexer latency is measured on a single NVIDIA H20 96GB GPU, while served TTFT is measured for a single request on $8\times$ H20 96GB GPUs with tensor parallelism 8, FP8 KV cache, and 8192-token prefill chunks. Further implementation and measurement details are given in Appendix G.

Downstream quality is evaluated on LongBench-v2 (Bai et al., 2025) and RULER (Hsieh et al., 2024) at context lengths up to 256K using the `lm-evaluation-harness` (Gao et al., 2024). We retain 407 of the 503 LongBench-v2 examples after excluding inputs beyond the 256K evaluation limit.

5.2 RETRIEVAL–EFFICIENCY TRADEOFF

Candidate recall measures the fraction of the exact Top-K set covered by the linear Top-B proposal. We additionally measure retained attention mass, i.e., the fraction of the original CSA attention mass on the exact Top-K compressed entries that is preserved by our final selection. Both metrics are paired with measured single-layer indexer latency on the 12-domain 64K V4 Diverse capture.

The Linear Indexer preserves high-attention entries at lower latency. Figure 6(a) shows that direct $B = 512$ retrieval covers 81.0% of the exact Top-512 set while retaining 91.3% of compressed-key attention mass. At $B = 2048$, these values reach 99.0% and 99.5%, respectively, while increasing to $B = 4096$ brings only marginal fidelity gains at substantially higher latency. Thus, increasing B mainly recovers the low-attention tail, at the cost of higher latency. Even direct $B = 512$ retrieval already preserves most of the attention mass, indicating that the highest-attention entries are largely retained. Additional retrieval and attention-fidelity results are provided in Appendix H.1.

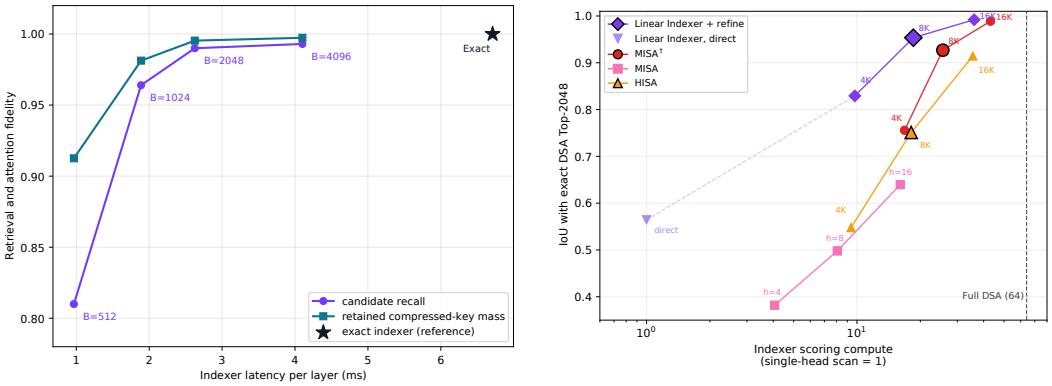


Figure 6: **(a)** Retrieval–efficiency tradeoff on DeepSeek-V4-Flash at 64K Diverse capture. Candidate recall and retained attention mass, normalized without the attention sink, are shown against indexer latency for different B; the star denotes the exact indexer. **(b)** Pareto frontier of proposal quality versus normalized indexer compute on DeepSeek-V3.2. Compute counts query–key scoring arithmetic only and is normalized so that one full single-head scan has cost 1. Labels denote the first-stage proposal budget B; black outlines mark $B = 8192$, and the dashed line denotes full 64-head DSA.

5.3 Comparison with alternative proposal mechanisms

We compare the Linear Indexer with MISA (Zhou et al., 2026) and HISA (Xu et al., 2026) on DeepSeek-V3.2. HISA filters the key axis at the block level, whereas MISA first pools keys into blocks and then routes each query to h of the 64 indexer heads. MISA† uses the 8-head router to propose Top-B candidates and reranks them with the original 64-head DSA scorer. All methods use the same query rows with more than 16,384 visible tokens and the same final budget $K = 2048$. Following MISA and HISA, we measure proposal quality by IoU with the exact Top-K set. Full details and results are provided in Appendix H.2.

The Linear Indexer pushes the retrieval–compute Pareto frontier. Figure 6(b) compares the tested proposal mechanisms under matched scoring compute. Compute is normalized so that one full single-head scan costs 1. This hardware-independent metric counts scoring operations while excluding preprocessing and selection; wall-clock latency on 64K capture is reported in Table 10 in Appendix H.2. The direct Linear Indexer achieves 0.564 IoU at cost 1.00, compared with 0.498 at 8.06 for MISA. At the common $B = 8192$ two-stage budget, the Linear Indexer reaches 0.954 IoU at cost 18.53, versus 0.927 at 25.59 for MISA†; HISA reaches 0.751 at a similar cost of 18.10. Thus, folding the weighted query heads into a single scan provides higher proposal quality than head routing or block-level key filtering under comparable compute.

5.4 Indexer and prefill efficiency

Figure 7 reports single-layer indexer and end-to-end prefill speedups on DeepSeek-V4-Flash at 64K–256K; full results and settings are provided in Appendix H.3.

Indexer acceleration translates into growing end-to-end prefill speedups at long context. Both Indexer and prefill speedups increase consistently with sequence length across budgets. Even at 8K context, the direct mode Linear Indexer with $B = 512$ is already 2.3× faster than the exact CSA Indexer (Fig. 15). At 256K, the Linear Indexer achieves 6.0×–10.4× indexer speedup, translating into a 1.33×–1.39× TTFT speedup. A batch-size-1 profile further shows that the CSA indexer share of prefill GPU busy time grows from 7.2% at 16K to 33.3% at 256K (Fig. 16). This explains why the end-to-end gains grow with context length while remaining smaller than the indexer-level gains.

5.5 Downstream quality on long-context benchmarks

We evaluate the exact CSA indexer and three Linear Indexer budgets, $B \in \{512, 1024, 2048\}$, on LongBench-v2 (Bai et al., 2025) and RULER (Hsieh et al., 2024) at context lengths up to 256K. Full evaluation protocols, confidence intervals, and task-level results are provided in Appendix H.4.

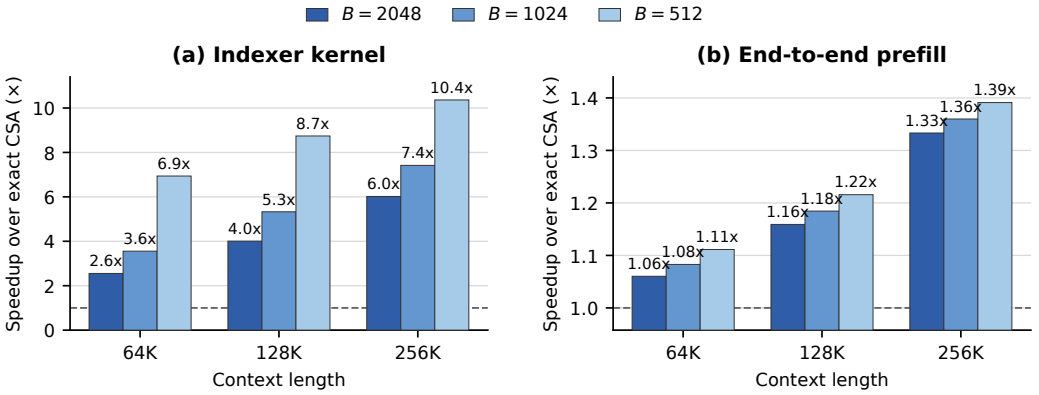


Figure 7: Speedup over exact CSA for **(a)** a single-layer indexer on one H20 and **(b)** single-request prefill TTFT on $8\times$H20 with TP8. Dashed lines mark $1\times$.

Table 1: Downstream quality under different candidate budgets. RULER reports the mean score over 13 tasks at each context length, with *Mean* averaging across the four context lengths; LongBench-v2 reports accuracy. Higher is better.

		RULER					LongBench-v2
Method	B	32K	64K	128K	256K	Mean	Acc.
Exact CSA	–	0.954	0.938	0.948	0.942	0.945	0.484
Linear Indexer	2048	0.952	0.934	0.948	0.944	0.944	0.499
	1024	0.959	0.943	0.951	0.944	0.949	0.484
	512	0.956	0.939	0.944	0.943	0.945	0.484

We do not additionally evaluate short-input benchmarks, since for any input with at most 512 indexed entries, even the smallest candidate budget selects all available entries and therefore produces exactly the same selected set as exact CSA.

The Linear Indexer preserves downstream quality across candidate budgets. LongBench-v2 accuracy remains between 0.484 and 0.499 across the three budgets, compared with .484 for exact CSA, while mean RULER scores remain between .944 and .949 versus .945. Notably, direct retrieval at $B = 512$ matches exact CSA on both LongBench-v2 (.484) and mean RULER (.945), and remains comparable at 256K (.943 vs. .942). Consequently, reducing the candidate budget from $B = 2048$ to the direct $B = 512$ setting produces no systematic downstream degradation through 256K context.

6 Limitations

Our mechanistic explanation is post hoc: we characterize learned structure that makes the linear score an effective retrieval proxy, but not how it emerges during training. Experiments are limited to DeepSeek-family Indexers up to 256K, so generalization to other learned indexers and longer contexts remains open. The Linear Indexer also does not accelerate decoding in our serving regime: it reduces indexer arithmetic but leaves the memory traffic unchanged. Finally, our Triton implementation is not fully optimized and does not represent the performance ceiling of the Linear Indexer.

7 Conclusion

We identify strong linear retrieval redundancy across three DeepSeek Indexer designs and trace it to learned head–weight structure. Exploiting this property, the training-free *Linear Indexer* removes the head-wise ReLUs and folds the weighted query heads exactly before the global scan. Our results show that this substantially reduces long-context indexing cost while preserving attention fidelity and downstream quality. These findings also motivate training sparse-attention indexers natively with the folded linear score, potentially reducing their long-context training cost.

Reproducibility Statement

We will release our analysis and kernel code. Detailed capture protocols, implementation settings, and evaluation procedures are provided in the appendix.

AI Use Statement

Generative AI tools were used at several stages of this work. The core idea of the Linear Indexer, the overall research direction, and the principal research hypotheses were conceived by the authors. AI tools subsequently assisted with preliminary empirical validation, feedback on research methodology and experimental design, and exploratory analysis and interpretation of experimental results. All reported statistics, figures, and conclusions were independently verified by the authors against the underlying data.

AI tools also assisted with the mathematical analysis. The authors formulated the mathematical claim and conceptual setup underlying the stylized toy model, after which AI tools assisted in deriving and drafting an initial proof. The authors subsequently checked, revised and polished the proof, its assumptions, and all resulting mathematical claims.

For implementation, AI tools generated portions of the implementation and analysis code, including parts of the Linear Indexer kernels, and assisted with debugging and performance optimization. The Linear Indexer algorithm and its conceptual design were specified by the authors. All AI-generated code was reviewed, executed, tested, and validated by the authors against the intended computation.

AI tools also assisted with code used to sample and assemble capture datasets from existing real datasets under author-specified protocols, as well as with figure design and preparation. No synthetic data generated by AI tools were used. AI tools were not used for translation.

Finally, AI tools assisted with literature search and related-work discovery, manuscript organization, drafting and revising portions of the paper and language editing. All cited sources and AI-assisted research suggestions, mathematical arguments, code, data-processing procedures, experimental analyses, figures, and manuscript text were reviewed by the authors. All reported numerical results were obtained by executing the authors' experimental pipelines. The authors take full responsibility for the final content, claims, code, and results of this work.

References

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 Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 3119–3137. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.172. URL https://doi.org/10.18653/v1/2024.acl-long.172.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 3639–3664. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.ACL-LONG.183. URL https://doi.org/10.18653/v1/2025.acl-long.183.

Yushi Bai, Qian Dong, Ting Jiang, Xin Lv, Zhengxiao Du, Aohan Zeng, Jie Tang, and Juanzi Li. Indexcache: Accelerating sparse attention via cross-layer index reuse. *CoRR*, abs/2603.12201, 2026. doi: 10.48550/ARXIV.2603.12201. URL https://doi.org/10.48550/arXiv.2603.12201.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *CoRR*, abs/2004.05150, 2020. URL https://arxiv.org/abs/2004.05150.

DeepSeek-AI. Deepseek-v3.2: Pushing the frontier of open large language models. *CoRR*, abs/2512.02556, 2025. doi: 10.48550/ARXIV.2512.02556. URL https://doi.org/10.48550/arXiv.2512.02556.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence. *CoRR*, abs/2606.19348, 2026. doi: 10.48550/ARXIV.2606.19348. URL https://doi.org/10.48550/arXiv.2606.19348.

DeepSeek-AI. Deepseek-v4.1-flash: Pushing the limits of kv cache compression. https://huggingface.co/deepseek-ai/DeepSeek-V4.1-Flash, 2026.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

GLM. GLM-5: from vibe coding to agentic engineering. *CoRR*, abs/2602.15763, 2026. doi: 10.48550/ARXIV.2602.15763. URL https://doi.org/10.48550/arXiv.2602.15763.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=kIoBbc76Sy.

Xiang Hu, Xinyu Wei, Hao Gu, Minshen Zhang, Tian Liang, Huayang Li, Lei Zhu, Yan Wang, Sirui Han, Yushi Bai, Kewei Tu, Haitao Mi, and Leo Liang. Hierarchical sparse attention done right: Toward infinite context modeling. *CoRR*, abs/2607.02980, 2026. doi: 10.48550/ARXIV.2607.02980. URL https://doi.org/10.48550/arXiv.2607.02980.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=rkgNKkHtvB.

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

Hong Liu, Yuan Cheng, Lin Niu, Yi Su, Yufei Xue, Anmin Liu, Guanghua Yu, and Jianchen Zhu. PIVOT: efficient query-group indexing for token-level sparse attention. *CoRR*, abs/2607.24593, 2026. doi: 10.48550/ARXIV.2607.24593. URL https://doi.org/10.48550/arXiv.2607.24593.

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, Zhiqi Huang, Huan Yuan, Suting Xu, Xinran Xu, Guokun Lai, Yanru Chen, Huabin Zheng, Junjie Yan, Jianlin Su, Yuxin Wu, Yutao Zhang, Zhilin Yang, Xinyu Zhou, Mingxing Zhang, and Jiezhong Qiu. Moba: Mixture of block attention for long-context llms. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/19eae75beed66321d62272e794a9c2ac-Abstract-Conference.html.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Trans. Assoc. Comput. Linguistics*, 9:53–68, 2021. doi: 10.1162/TACL_A_00353. URL https://doi.org/10.1162/tacl_a_00353.

Xinyu Shi, Kairong Luo, Zhen Zheng, and Wenguang Chen. Robsa: Rope-based blockwise sparse multi-head latent attention. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2026, San Diego, California, United States, July 2-7, 2026*, pp. 1028–1044. Association for Computational Linguistics, 2026. doi: 10.18653/V1/2026.ACL-LONG.46. URL https://doi.org/10.18653/v1/2026.acl-long.46.

Tencent Hy Team. Hy4 preview, 2026. URL https://github.com/Tencent-Hunyuan/Hy4-preview. Official model release and technical documentation.

Philippe Tillet, Hsiang-Tsung Kung, and David D. Cox. Triton: an intermediate language and compiler for tiled neural network computations. In Tim Mattson, Abdullah Muzahid, and Armando Solar-Lezama (eds.), *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages, MAPL@PLDI 2019, Phoenix, AZ, USA, June 22, 2019*, pp. 10–19. ACM, 2019. doi: 10.1145/3315508.3329973. URL https://doi.org/10.1145/3315508.3329973.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

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 Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024a. URL http://papers.nips.cc/paper_files/paper/2024/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024b. URL https://openreview.net/forum?id=NG7sS51zVF.

Yufei Xu, Fanxu Meng, Fan Jiang, Yuxuan Wang, Ruijie Zhou, Zhaohui Wang, Jiexi Wu, Zhixin Pan, Xiaojuan Tang, Wenjie Pei, Tongxuan Liu, Di Yin, Xing Sun, and Muhan Zhang. Hisa: Efficient hierarchical indexing for fine-grained sparse attention. In *Conference on Language Modeling (COLM)*, 2026.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 23078–23097. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.ACL-LONG.1126. URL https://doi.org/10.18653/v1/2025.acl-long.1126.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/c8512d142a2d849725f31a9a7a361ab9-Abstract.html.

Wen Zan, Jiaqi Zhang, Jianchao Tan, Hong Liu, Cunguang Wang, Xiang Li, Duyue Ma, Guanyu Wu, Yifan Lu, Fengcun Li, Yerui Sun, Peng Pei, Yuchen Xie, and Xunliang Cai. Longcat sparse attention: Taming the lightning via streaming-aware hierarchical cross-layer indexing, 2026. URL https://arxiv.org/abs/2608.01662.

Weilin Zhao, Zihan Zhou, Zhou su, Chaojun Xiao, Yuxuan Li, Yanghao Li, Yudi Zhang, Weilun Zhao, Zhen Li, Yuxiang Huang, Ao Sun, Xu Han, and Zhiyuan Liu. InfLLM-v2: Dense-sparse switchable attention for seamless short-to-long adaptation. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=ZzF9V0H6Vi.

Yu Zhao, Zekun Zhang, Fan Jiang, Bo Zeng, Linlong Xu, Shimin Shan, Yu Liu, Longyue Wang, and Weihua Luo. LISA: linear-indexed sparse attention for efficient long-context reasoning. *CoRR*, abs/2607.19358, 2026b. doi: 10.48550/ARXIV.2607.19358. URL https://doi.org/10.48550/arXiv.2607.19358.

Ruijie Zhou, Fanxu Meng, Yufei Xu, Tongxuan Liu, Guangming Lu, Muhan Zhang, and Wenjie Pei. MISA: mixture of indexer sparse attention for long-context LLM inference. *CoRR*, abs/2605.07363, 2026. doi: 10.48550/ARXIV.2605.07363. URL https://doi.org/10.48550/arXiv.2605.07363.

A Related Work

Sparse attention. Longformer and BigBird combine local attention with sparse global or random connections (Beltagy et al., 2020; Zaheer et al., 2020). Reformer and Routing Transformer perform content-dependent routing via locality-sensitive hashing or online clustering (Kitaev et al., 2020; Roy et al., 2021). StreamingLLM retains attention sinks and a recent-token window for bounded-memory streaming (Xiao et al., 2024b), while InfLLM retrieves blocks from an auxiliary context memory (Xiao et al., 2024a). RoBSA takes a training-free approach for MLA, using RoPE-only attention scores as a lightweight proxy for block selection before exact sparse attention (Shi et al., 2026). More recent sparse-attention systems include Native Sparse Attention (Yuan et al., 2025), MoBA (Lu et al., 2025), InfLLM-V2 (Zhao et al., 2026a), and HiLS-Attention (Hu et al., 2026). DeepSeek-V3.2 introduces DeepSeek Sparse Attention and its multi-head ReLU Lightning Indexer (DeepSeek-AI, 2025); DeepSeek-V4 applies the indexer over compressed entries in CSA (DeepSeek-AI, 2026), and GLM-5 adopts a related learned indexing paradigm (GLM, 2026). LISA combines a trainable linear-attention branch with sparse self-attention whose tokens are selected by a learned multi-head ReLU Indexer (Zhao et al., 2026b). Despite the similar terminology, its linear component is an attention branch rather than a linearization of the Indexer itself.

Indexer acceleration. Recent work specifically targets the cost of learned sparse-attention indexing. HISA reduces the key axis through hierarchical block filtering before token-level refinement (Xu et al., 2026), while MISA reduces the head axis by routing each query to a subset of indexer heads and additionally studies hierarchical re-ranking (Zhou et al., 2026). IndexCache reuses selected indices across layers (Bai et al., 2026); PIVOT amortizes prefix traversal across nearby queries and optionally performs per-query refinement (Liu et al., 2026); and LongCat Sparse Attention combines hierarchical selection, cross-layer reuse, and a streaming-oriented memory layout (Zan et al., 2026). In a concurrent release, DeepSeek-V4.1 introduced CSA2, which further combines these axes by sharing Indexer keys and Top-K selections across layers and restricting Reindex operations to a hierarchical candidate pool (DeepSeek-AI, 2026).

B A Stylized Toy Model for ReLU-Induced Correlation

The following result is a deliberately stylized thought experiment rather than a probabilistic model of the trained indexer. Its purpose is to show that the ReLU decomposition intrinsically favors correlation with the signed linear component when the attention score is Gaussian random initialized.

Proposition B.1 (Linear dominance in a symmetric toy model)**.** *Let $x_1, \dots, x_H$ be independent random variables with zero mean, symmetric distributions, and finite second moments. Let $w_1, \dots, w_H$ be deterministic coefficients, and define*

$$I = \sum_{h=1}^{H} w_h \,\mathrm{ReLU}(x_h), \qquad L = \sum_{h=1}^{H} w_h x_h, \qquad A = \sum_{h=1}^{H} w_h |x_h|. \tag{6}$$

Assume $\mathrm{Var}(L) > 0$ *and* $\mathrm{Var}(A) > 0$*. Then*

$$I = \frac{L + A}{2}, \qquad \mathrm{Cov}(L, A) = 0, \tag{7}$$

and

$$\mathrm{Corr}(I, L) = \sqrt{\frac{\mathrm{Var}(L)}{\mathrm{Var}(L) + \mathrm{Var}(A)}}, \tag{8}$$

$$\mathrm{Corr}(I, A) = \sqrt{\frac{\mathrm{Var}(A)}{\mathrm{Var}(L) + \mathrm{Var}(A)}}. \tag{9}$$

Moreover,

$$\mathrm{Corr}(I, L) > \frac{1}{\sqrt{2}} > \mathrm{Corr}(I, A). \tag{10}$$

Proof. The identity

$$\mathrm{ReLU}(x) = \frac{x + |x|}{2}$$

immediately gives $I = (L + A)/2$.

Because the distribution of each x_h is symmetric, $x_h|x_h|$ is an odd function and therefore

$$\mathbb{E}[x_h|x_h|] = 0.$$

Since $\mathbb{E}[x_h] = 0$, this implies

$$\mathrm{Cov}(x_h, |x_h|) = 0.$$

For $h \neq h'$, independence and $\mathbb{E}[x_h] = 0$ give

$$\mathrm{Cov}(x_h, |x_{h'}|) = 0.$$

Hence

$$\mathrm{Cov}(L, A) = \sum_{h=1}^{H} \sum_{h'=1}^{H} w_h w_{h'} \,\mathrm{Cov}(x_h, |x_{h'}|) = 0.$$

Using $I = (L + A)/2$ and $\mathrm{Cov}(L, A) = 0$, we obtain

$$\begin{aligned}
\mathrm{Var}(I) &= \frac{1}{4}\left(\mathrm{Var}(L) + \mathrm{Var}(A)\right), \\
\mathrm{Cov}(I, L) &= \frac{1}{2}\,\mathrm{Var}(L), \\
\mathrm{Cov}(I, A) &= \frac{1}{2}\,\mathrm{Var}(A).
\end{aligned}$$

Substituting these identities into the definition of correlation yields eqs. (8) and (9).

It remains to compare the two component variances. Independence gives

$$\begin{aligned}
\mathrm{Var}(L) &= \sum_{h=1}^{H} w_h^2 \,\mathrm{Var}(x_h), \\
\mathrm{Var}(A) &= \sum_{h=1}^{H} w_h^2 \,\mathrm{Var}(|x_h|).
\end{aligned}$$

For each h,

$$\mathrm{Var}(|x_h|) = \mathbb{E}[x_h^2] - \mathbb{E}[|x_h|]^2 \leq \mathbb{E}[x_h^2] = \mathrm{Var}(x_h),$$

with strict inequality whenever $\mathrm{Var}(x_h) > 0$. Since $\mathrm{Var}(L) > 0$, at least one term with $w_h \neq 0$ has positive variance. Hence

$$\mathrm{Var}(A) < \mathrm{Var}(L).$$

Together with eqs. (8) and (9), this immediately implies

$$\mathrm{Corr}(I, A) < \frac{1}{\sqrt{2}} < \mathrm{Corr}(I, L).$$

□

Gaussian special case. If each x_h is zero-mean Gaussian, not necessarily with the same variance, then

$$\mathrm{Var}(|x_h|) = \left(1 - \frac{2}{\pi}\right) \mathrm{Var}(x_h).$$

Consequently,

$$\mathrm{Var}(A) = \left(1 - \frac{2}{\pi}\right) \mathrm{Var}(L),$$

and the two correlations are exactly

$$\mathrm{Corr}(I, L) = \sqrt{\frac{\pi}{2(\pi - 1)}} \approx 0.856,$$

$$\mathrm{Corr}(I, A) = \sqrt{\frac{\pi - 2}{2(\pi - 1)}} \approx 0.516.$$

Thus, even in an untrained Gaussian random initialization, the exact ReLU score is substantially more correlated with its signed linear component than with its nonlinear absolute-value component.

C Capture Datasets and Protocol

C.1 Pilot capture

The main mechanistic analysis uses six 64K prompts constructed from non-synthetic LongBench tasks (Bai et al., 2024): `narrativeqa`, `qasper`, `qmsum`, `multi_news`, `hotpotqa`, and `repobench-p`. These tasks cover narrative text, scientific documents, meetings, news, multi-document question answering, and repository-level code. For each task, records are deterministically shuffled and concatenated without replacement until reaching 65,536 model tokens. We use neither repeated documents nor synthetic padding.

Each prompt contributes 128 causal query positions from the second half of the context. The samples combine broad coverage of $[32\mathrm{K}, 64\mathrm{K})$ with denser sampling near the context tail, 4-token CSA compression boundaries, and 128-token Heavily Compressed Attention (HCA) compression boundaries. Restricting queries to the second half ensures that every sampled row observes at least 8,192 compressed entries under the V4 compression ratio of four, so Top-512 retrieval statistics are well defined throughout.

C.2 Capture inventory

Beyond the pilot set, we construct additional captures to test generalization across domains, context lengths, checkpoints, and indexer architectures. Table 2 summarizes the capture sets used in the paper.

The Diverse set spans 12 domains from LongBench (Bai et al., 2024) covering books, scientific papers, meetings, news, Wikipedia QA, government reports, dialogue, Chinese QA, Chinese legal text, code completion, repository-level code, and mathematics. Prompts are constructed independently at each context length using the same non-repeated-record principle as the pilot. The V3.2 capture uses the same domain and length grid but records all 61 token-level DSA indexer layers with native retrieval budget $K = 2048$. The Base capture reruns the pilot prompts under the V4-Flash-Base checkpoint using the same capture configuration. The V4.1 capture uses the same 12 domains at 64K context and records all eight CSA2 scorer layers with native budget $K = 512$: four full-scan source layers, including the candidate-source layer L20, and four hierarchical reindex layers.

C.3 Inference and captured tensors

V4 captures are collected with tensor parallelism eight, eager execution, FP8 MLA KV cache, and prefix caching disabled. DeepSeek-V4.1-Flash captures use a separate vLLM build at commit `c9d909e802a3`, which provides native V4.1 support. We add non-intrusive hooks only to export CSA2 intermediate states; they do not modify model computation or retrieval decisions. For every sampled query and indexer layer, we record the pre- and post-RoPE indexer queries, the 64 query-dependent head weights, the valid post-RoPE indexer-key cache, and the production nonlinear indexer scores and Top-512 indices. All analyses restrict both exact and linear scores to the same causal valid-key range. The pilot capture therefore contains $6 \times 128 \times 21 = 16{,}128$ query–layer instances.

Attention-fidelity analysis uses the same 12 prompts as the 64K slice of the Diverse set, with 32 causal query rows per prompt. For these rows, we additionally retain the attention states required to reconstruct the exact and Linear-Indexer CSA outputs. Generalization statistics in Appendix E use the same query-sampling protocol across domains, lengths, and models.

Table 2: Capture sets used in this paper. "Files" counts layer × prompt capture files, each containing the sampled query rows. The 64K Diverse prompts are also used for attention-fidelity analysis.

Set	Model	Prompts	Lengths	Layers	Files	Role
Pilot	V4-Flash-0731	6 LongBench tasks	64K	21	126	Main analysis
Diverse	V4-Flash-0731	12 domains	4K–64K	21	1,260	Generalization; fidelity
Base	V4-Flash-Base	Pilot 6	64K	21	126	Checkpoint
V3.2	DeepSeek-V3.2	12 domains	4K–64K	61	3,660	Cross-model
V4.1	V4.1-Flash	12 domains	64K	8	96	Cross-model

D Additional Mechanism Evidence

D.1 Weight-pairing intervention, per layer

To test whether the learned head–weight pairing is necessary for high linear retrieval fidelity, we keep the trained q and k fixed and perturb only w. We consider two controls: permuting weights across heads, which preserves the per-query weight multiset, and randomizing weight signs while preserving $|w|$. The left panel of Fig. 8 evaluates each perturbed linear score against the original trained Top-512, while the right panel evaluates it against the nonlinear indexer formed with the same perturbed weights.

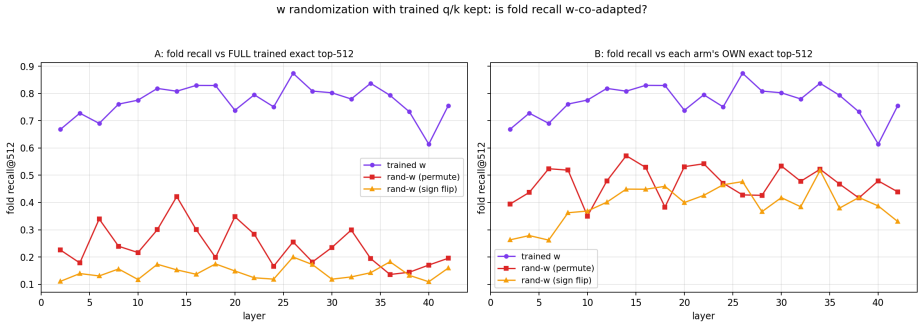


Figure 8: The intervention of section 3.4 resolved per layer. Trained q and k are kept intact, preserving the key geometry and the PD/ND sign structure; only the learned head–weight pairing is destroyed. Left is the task target, right each arm's own exact score. The collapse is uniform across layers, so it is not driven by a few unusual layers.

Destroying the pairing sharply degrades linear retrieval under both criteria. Mean task recall falls from 0.771 to approximately 0.24 under weight permutation and 0.14 under sign randomization; even against each perturbed arm's own nonlinear target, recall remains only about 0.47 and 0.39. The degradation occurs across essentially all layers. Thus, stable head signs alone are insufficient: high linear retrieval fidelity depends on how the learned query-dependent weights align oppositely oriented heads.

D.2 PD–ND contribution agreement

This appendix specifies the metrics and controls used in Fig. 5 and Fig. 9. PD and ND identities are fixed using the overall positivity rate defined in Sec. 3.3 and are not recomputed for individual queries. We suppress the layer index below.

For each query t and group $G \in \{\mathrm{PD}, \mathrm{ND}\}$, define

$$v_t^G := \sum_{h \in G} w_{t,h} q_{t,h}, \qquad C_{t,s}^G := (v_t^G)^\top k_s = \sum_{h \in G} w_{t,h} q_{t,h}^\top k_s.$$

All statistics are computed independently for each query–layer instance over its S_t valid compressed keys. The evaluation contains 6 prompts × 21 CSA layers × 128 sampled queries $= 16{,}128$ such instances.

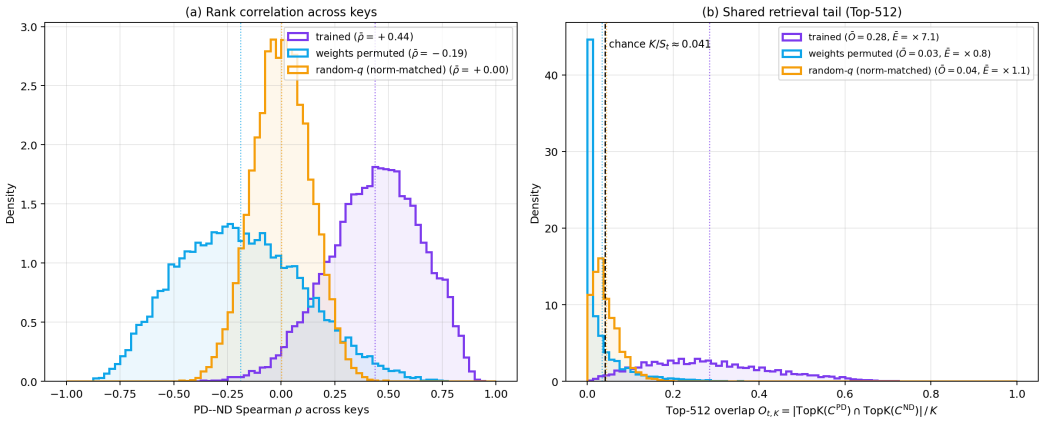


Figure 9: Agreement between PD and ND contributions. **(a)** Per-query Spearman rank correlations over valid keys. **(b)** Overlap between the selected PD and ND Top-512 sets. The black dashed line marks the approximate random selection overlap, and colored dotted lines mark distribution means.

Score correlation. For each query, we compute the Pearson correlation between the PD and ND contributions across valid keys:

$$r_t = \mathrm{Corr}_s \left(C_{t,s}^{\mathrm{PD}}, C_{t,s}^{\mathrm{ND}} \right).$$

In Fig. 5(a), the displayed query–layer instance is chosen mechanically as the one whose r_t is closest to the global median. Red points indicate the Top-512 key entries selected by the exact nonlinear Indexer.

Rank correlation and Top-K overlap. We additionally compute

$$\rho_t = \mathrm{Spearman}_s \left(C_{t,s}^{\mathrm{PD}}, C_{t,s}^{\mathrm{ND}} \right)$$

and, for $K = 512$,

$$O_{t,K} = \frac{\left| \mathrm{TopK}(C_t^{\mathrm{PD}}) \cap \mathrm{TopK}(C_t^{\mathrm{ND}}) \right|}{K}, \qquad E_{t,K} = \frac{O_{t,K}}{K/S_t}.$$

Here, $O_{t,K}$ is the observed Top-K overlap, while K/S_t is the expected overlap for two independent random Top-K sets. Thus, $E_{t,K}$ measures how many times the observed overlap exceeds the random expectation: $E_{t,K} = 1$ corresponds to chance level. Because the number of valid keys S_t varies across queries, the random overlap baseline K/S_t also varies. We therefore compute $E_{t,K} = O_{t,K}/(K/S_t)$ separately for each query and then average $E_{t,K}$ across queries.

Controls. The weight-permutation control follows Appendix D.1. For the random query-direction baseline, each trained query is replaced by a norm-matched isotropic random direction while keeping the trained weights, keys, and PD/ND assignments fixed.

E Generalization of Linear Retrieval and Learned Structure

This appendix tests whether the linear retrieval behavior and learned structure identified in section 3 persist across domains, context lengths, checkpoints, and models. All capture sets are documented in Appendix C. To characterize the learned structural patterns identified above, we compute the following six statistics for each capture file on a fixed random 32-row subset of the 128 captured query rows, and then average them across files.

First, *direct linear recall* measures how much of the exact Top-K set is recovered by the linear Top-K:

$$R_{\mathrm{direct}} = \frac{|\mathrm{TopK}(L_t) \cap \mathrm{TopK}(I_t)|}{K}.$$

Second, *key concentration*

$$C_{\text{cone}} = \Pr_s\left[k_s^\top \mu_k > 0\right]$$

measures the fraction of keys lying in the positive half-space defined by their mean direction μ_k.

Third, we measure how strongly a head's alignment with the mean key direction predicts the sign of its query–key products. Let

$$a_h = \mathbb{E}_t[\cos(q_{t,h}, \mu_k)]\,, \qquad p_h = \Pr_{t,s}\left(q_{t,h}^\top k_s > 0\right)\,,$$

and report their Pearson correlation $\rho_{a,p}$ across heads.

Fourth, we report the fraction of heads classified as PD or ND,

$$P_{\text{PD/ND}} = \Pr_h[p_h < 0.1 \text{ or } p_h > 0.9]\,.$$

Fifth, *opposite weight signs* measures how often the learned weight sign opposes the dominant query–key product sign among PD/ND heads:

$$A_{\text{wp}} = \Pr_h[\text{sign}(\widetilde{w}_h) = -\,\text{sign}(2p_h - 1) \mid h \in \text{PD} \cup \text{ND}]\,, \qquad \widetilde{w}_h = \frac{\mathbb{E}_t[w_{t,h}]}{\mathbb{E}_t[|w_{t,h}|]}.$$

Finally, we measure whether PD and ND contributions rank keys similarly using their mean per-query Spearman correlation,

$$\rho_{\text{PD,ND}} = \mathbb{E}_t\left[\text{Spearman}_s\left(C_{t,s}^{\text{PD}}, C_{t,s}^{\text{ND}}\right)\right].$$

PD/ND labels use the same thresholds as in section 3.3. Every generalization experiment below reports this same set of six statistics.

E.1 Domains and context lengths

Table 3: Mechanism signatures by context length on V4, each cell averaged over 12 domains and 21 CSA layers.

Length	4K	8K	16K	32K	64K
R_{direct}	0.933	0.877	0.850	0.821	0.809
C_{cone}	0.9975	0.9985	0.9990	0.9993	0.9996
$\rho_{a,p}$	0.949	0.948	0.948	0.951	0.952
$P_{\text{PD/ND}}$	0.450	0.449	0.441	0.441	0.440
A_{wp}	0.692	0.744	0.779	0.807	0.818
$\rho_{\text{PD,ND}}$	+0.291	+0.319	+0.364	+0.396	+0.408

Table 3 separates two trends that a recall-only view conflates. Direct recall decreases smoothly with length ($0.933 \to 0.809$), yet the learned structural patterns either remain stable (key concentration, axis–sign correlation, PD/ND fraction) or strengthen (opposite weight signs $+0.13$, PD–ND rank correlation $+0.12$). This is consistent with a harder retrieval problem at longer contexts—more candidates must be ranked for each query—rather than a weakening of the learned structure.

Figure 10 further shows that this trend is consistent across domains: at any fixed length, the 12 domains differ by at most 0.04 in direct Top-512 recall. Figure 11 provides a complementary layer-wise view. Despite substantial variation across layers, the 12 domains follow highly similar profiles for both direct linear recall and the PD/ND-head fraction, indicating that these patterns are not driven by a small subset of domains.

E.2 The learned structure precedes post-training

Table 4 compares the Base and post-trained V4-Flash checkpoints on the same six 64K prompts. All six mechanism signatures are already pronounced in the Base model: direct linear recall is 0.782, key concentration is 0.9999, and the alignment–sign correlation reaches 0.953. These values remain similar after post-training, with direct recall of 0.770 and an alignment–sign correlation of 0.955. These results suggest that linear retrieval redundancy emerges during pre-training and persists through post-training.

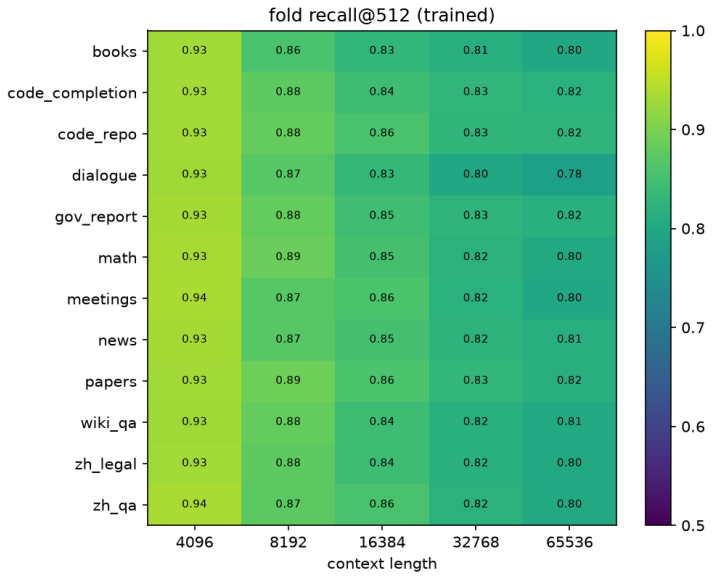


Figure 10: Direct linear Top-512 recall by domain and context length on V4, averaged over 21 CSA layers. Variation is dominated by context length; within each length, the 12 domains differ by at most 0.04.

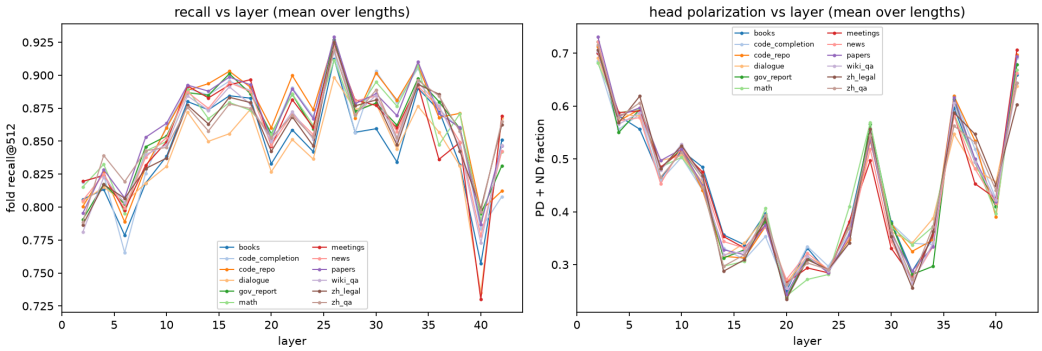


Figure 11: Layer-wise consistency across domains on V4, averaged over context lengths. **(a)** Direct linear Top-512 recall. **(b)** Fraction of PD/ND heads. Across both metrics, the 12 domains follow similar layer-wise profiles despite substantial variation across layers.

Table 4: Mechanism signatures across V4-Flash checkpoints, computed on the six matched 64K prompts.

Signature	Base	0731
R_{direct}	0.782	0.770
C_{cone}	0.9999	0.9996
$\rho_{a,p}$	0.953	0.955
$P_{\text{PD/ND}}$	0.466	0.451
A_{wp}	0.760	0.820
$\rho_{\text{PD,ND}}$	+0.401	+0.432

E.3 CROSS-MODEL GENERALIZATION

We further test whether the learned structure identified on V4-Flash generalizes to two substantially different Indexer designs: token-level DSA in DeepSeek-V3.2 and hierarchical CSA2 in DeepSeek-V4.1-Flash. Table 5 summarizes the same six mechanism signatures on matched 12-domain, 64K-context captures.

Table 5: Architecture settings and mechanism signatures across three learned Indexer designs on matched 12-domain, 64K-context captures. The V4.1 full-scan group contains 2:1-compressed layers L2–L14 and the uncompressed candidate-source layer L20.

	V4-Flash	V3.2	V4.1-Flash		
	CSA	DSA	All	Full scan	Reindex
Architecture					
Indexer heads H	64	64	32	32	32
Key compression	4:1	1:1	-	2:1/1:1	1:1
Num Indexer layers	21	61	8	4	4
Index cache interval	1	1	-	6	4
Top-K	512	2048	512	512	512
Mechanism signatures					
R_{direct}	0.809	0.732	0.691	0.712	0.670
C_{cone}	0.9996	1.0000	0.9998	0.9999	0.9997
$\rho_{a,p}$	0.952	0.875	0.961	0.958	0.964
$P_{\text{PD/ND}}$	0.440	0.620	0.643	0.657	0.628
A_{wp}	0.818	0.835	0.757	0.750	0.765
$\rho_{\text{PD,ND}}$	+0.408	+0.481	+0.511	+0.521	+0.501

DeepSeek-V3.2. The same structure remains visible across all 61 DSA Indexer layers. At the native budget, direct linear recall is 0.732, compared with 0.809 on V4-Flash. This difference is consistent with retrieving 2,048 keys from roughly 64K token-level candidates in V3.2, rather than 512 keys from roughly 16K compressed entries in V4. The remaining signatures are comparable to or stronger than those observed on V4, including stable head signs, opposite weight signs, and similar PD/ND key rankings. Figure 12 further shows that the layer-wise profiles are consistent across domains.

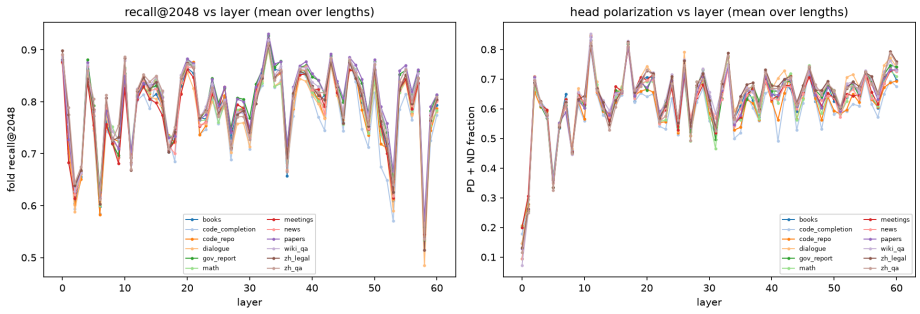


Figure 12: Layer-wise consistency across domains on DeepSeek-V3.2, averaged over five context lengths. **(a)** Direct linear Top-2048 recall. **(b)** Fraction of PD/ND heads. The 12 domains follow similar profiles across all 61 DSA Indexer layers.

DeepSeek-V4.1-Flash. CSA2 provides a further test under hierarchical selection and cross-layer reuse. We analyze its eight scorer layers: four full-scan layers (L2–L20) and four hierarchical reindex layers (L24–L36); reuse-only layers do not invoke an Indexer and are therefore omitted. Direct linear Top-512 recall averages 0.691 across these layers, with 0.712 on full-scan layers and 0.670 on reindex layers. The lower reindex average is primarily caused by L24, where recall drops to 0.489 at the transition to hierarchical reindexing; it recovers to above 0.72 in the subsequent reindex layers. The other five mechanism signatures differ by at most 0.03 between the two groups. For reindex layers,

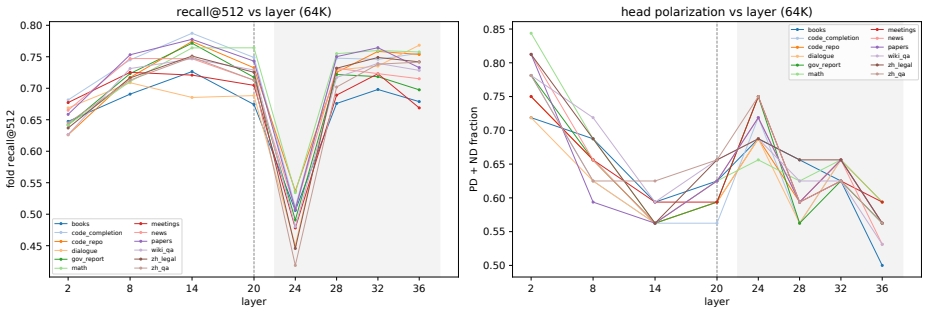


Figure 13: Layer-wise consistency across domains on DeepSeek-V4.1-Flash at 64K context. **(a)** Direct linear Top-512 recall. **(b)** Fraction of PD/ND heads. The dashed line marks the final full-scan layer, and the shaded region denotes hierarchical reindex layers.

we retain the production hierarchical candidate pool obtained in L20 to isolate the effect of linearizing the scorer. As shown in Figure 13, the domain-wise curves remain closely aligned in both regimes. These results extend the learned structure from token-level DSA and compressed CSA to hierarchical CSA2. Even several layers share an indexer, they still retain a high recall and the same structure.

F Linear Indexer Pseudocode

Algorithm 1 summarizes the complete inference procedure. Besides the linear proposal and exact refinement described in section 4, the deployed implementation includes a length-dependent fallback: for each candidate budget B, we enable the Linear Indexer only when the number of visible keys S exceeds a measured crossover $S_{\min}(B)$. Below this threshold, the original nonlinear indexer is used unchanged. The crossover values and their system measurements are reported in Appendix G.3.

Algorithm 1 Linear Indexer for one query

Require: Query heads $\{q_{t,h}\}_{h=1}^{H}$, weights $\{w_{t,h}\}_{h=1}^{H}$, keys $\{k_s\}_{s=1}^{S}$, budgets $B \geq K$, crossover $S_{\min}(B)$

if $S \leq S_{\min}(B)$ **then**
  Compute $I_{t,s} = \sum_{h=1}^{H} w_{t,h}\,\mathrm{ReLU}(q_{t,h}^{\top} k_s)$ for all s
  return $\mathrm{TopK}_s(I_{t,s})$
end if
$\bar{q}_t \leftarrow \sum_{h=1}^{H} w_{t,h} q_{t,h}$
$L_{t,s} \leftarrow \bar{q}_t^{\top} k_s$ for all s
$\mathcal{C}_B(t) \leftarrow \mathrm{TopB}_s(L_{t,s})$
if $B = K$ **then**
  return $\mathcal{C}_K(t)$
end if
Compute $I_{t,s} = \sum_{h=1}^{H} w_{t,h}\,\mathrm{ReLU}(q_{t,h}^{\top} k_s)$ for $s \in \mathcal{C}_B(t)$
return $\mathrm{TopK}_{s \in \mathcal{C}_B(t)}(I_{t,s})$

The procedure therefore has three execution regimes. For short prefixes it falls back to the original indexer; for $B = K$, it uses the folded linear score directly without refinement; and for $B > K$, it uses linear proposal followed by exact nonlinear re-scoring on the proposed subset. The threshold $S_{\min}(B)$ is a systems parameter rather than a model parameter and is selected from measured kernel crossovers without retraining.

G Implementation Details

G.1 Runtime configuration

All served experiments use vLLM 0.25.0 (Kwon et al., 2023), PyTorch 2.11, Triton 3.7 (Tillet et al., 2019), and CUDA 13.0 on 8× NVIDIA H20 96 GB GPUs with tensor parallelism 8. DeepSeek-V4-Flash uses its native quantized MoE configuration and an `fp8_ds_mla` KV cache. Chunked prefill is enabled with 8,192-token chunks and a maximum model length of approximately 256K. CUDA graphs are enabled for reported served TTFT measurements. Eager execution is used for activation capture, kernel validation, and the separate prefill-time composition profile in Fig. 16. Speculative decoding and prefix caching are disabled. Quality evaluations use greedy decoding with a fixed engine seed.

G.2 Linear Indexer pipeline

The Linear Indexer consists of three stages: query folding and linear scan, Top-B selection, and optional exact refinement. We describe the prefill implementation used for all reported latency measurements.

1. Query folding and linear scan. The indexer shares one key vector across its $H = 64$ heads, allowing the linear score to be evaluated through the folded query

$$\bar{q}_t = \sum_{h=1}^{H} w_{t,h} q_{t,h}, \qquad L_{t,s} = \bar{q}_t^{\top} k_s.$$

Folding and scanning are implemented as two Triton kernels. The folding kernel partitions the $d = 128$ dimension into 32-dimensional shards and reduces the 64 heads in groups of eight using four warps. It accumulates in FP32 but writes the folded query to an FP16 buffer. The scan kernel streams keys directly from the FP8 indexer cache and evaluates the query–key products with tensor-core MMA and FP32 accumulation, without materializing the original $H \times S$ head–key score tensor.

2. Top-B selection. The linear scores are reduced to $\mathcal{C}_B(t)$ with `top_k_per_row_prefill`, the radix-select Top-k CUDA kernel exposed through `vllm._custom_ops` and used by the production indexer in `vllm.model_executor.layers.sparse_attn_indexer`. The same selector is reused after refinement for the final Top-K, avoiding differences between the exact and linear paths due to Top-k implementation.

3. Exact refinement. For $B > K$, a fused Triton kernel gathers the B shortlisted keys and evaluates the original nonlinear indexer score

$$I_{t,s} = \sum_{h=1}^{H} w_{t,h} \,\mathrm{ReLU}(q_{t,h}^{\top} k_s), \qquad s \in \mathcal{C}_B(t).$$

Candidate gathering and 64-head scoring are performed in the same kernel: each tile dereferences the candidate indices directly, so gathered keys are never materialized. The cached FP8 keys and queries are fed to the MMA unwidened, performing the $d = 128$ reductions as FP8 MMA with FP32 accumulation; the ReLU, the head weighting, and the dequantization scale are then applied in FP32. The final K entries are selected with the same production Top-k kernel described above.

G.3 Budget-dependent dispatch

At short contexts, the overhead outside the global scan can outweigh the benefit of the Linear Indexer. For each candidate budget B, we therefore use a measured crossover threshold $S_{\min}(B)$: the original nonlinear indexer is used when $S \le S_{\min}(B)$, and the Linear Indexer is dispatched otherwise. Here S denotes the number of visible compressed CSA entries, rather than the number of original tokens. Under the V4 compression ratio of 4, the deployed thresholds are shown in Table 6.

The crossover moves to shorter contexts as B decreases because the shortlist and refinement costs become smaller; at $B = K = 512$, refinement disappears entirely. We therefore set the budget-dependent dispatch threshold S_{min} to the measured crossover point in the single-layer latency sweep

Table 6: Budget-dependent dispatch thresholds on H20. The thresholds are measured crossover points rather than learned model hyperparameters.

B	Mode	$S_{\min}$ (entries)	Approx. tokens
512	direct linear	0	0
1024	linear + exact refinement	2,048	8K
2048	linear + exact refinement	4,096	16K

of Fig. 15. With chunked prefill, dispatch is based on the visible prefix of each chunk, so early chunks of a long request may still use the original indexer.

G.4 Measurement protocol

Indexer latency is measured on a single H20 with 8,192 query rows per layer. For each (B, S) setting, we run 3 warm-up iterations followed by 20 timed iterations and report the median CUDA-event time. Served TTFT is measured with a single request on $8\times$ H20 GPUs with TP size 8. Each point is the median of three trials using identical prompts across compared methods. Prefix-cache reuse is prevented between paired measurements. TTFT is measured from request submission to the first generated token.

H Detailed Experimental Results

This appendix provides additional experimental results supporting section 5, including retrieval and attention fidelity, proposal comparisons, kernel/prefill measurements and detailed downstream evaluations.

H.1 Retrieval and attention fidelity

We first provide complementary fidelity results for the candidate-budget analysis in section 5.2. On the six-prompt V4 pilot capture, direct linear Top-512 recovers 0.771 of the exact Top-512 set. Increasing only the proposal budget raises mean coverage to 0.989 at $B = 2048$, while the fifth percentile improves from 0.621 to 0.947 (Tab. 7). The same trend appears in the token-level DSA indexer of DeepSeek-V3.2: fourfold over-retrieval reaches 0.980 mean coverage and eightfold reaches 0.997, indicating that high-recall linear proposals are not specific to compressed V4 attention.

Set recall treats every selected entry equally, although their contributions to the subsequent attention computation can differ substantially. We therefore reconstruct CSA attention on the same 12 prompts as the 64K Diverse capture.We measure the fraction of exact compressed-key attention mass retained by the Linear Indexer's final selection, excluding the unchanged sink from the normalization, together with the resulting attention-output discrepancy. As shown in Table 8, attention fidelity saturates faster than set recall. At $B = 1024$, 96.4% set recall retains 98.1% of the compressed-key attention mass and yields output cosine 0.9998; at $B = 2048$, these values reach 99.0%, 99.5%, and 1.0000, respectively.

Table 7: Candidate coverage on the V4 pilot capture at 64K and on the matched DeepSeek-V3.2 capture. V4 statistics use 16,128 query–layer instances; V3.2 averages over 12 domains and context lengths from 4K–64K.

	V4-Flash, $K = 512$						V3.2, $K = 2048$		
B	512	768	1024	1536	2048	B/K	Mean	64K	Worst layer
Mean	0.771	0.893	0.942	0.978	0.989	1	0.790	0.732	0.544
p5	0.621	0.744	0.822	0.906	0.947	2	0.940	0.915	0.716
p1	0.529	0.645	0.730	0.832	0.895	4	0.980	0.980	0.875
						8	0.997	0.997	0.970

Table 8: Attention fidelity on the 64K V4 Diverse capture. "Mass" is the fraction of exact compressed-key softmax mass retained by the Linear Indexer's final Top-K selection, normalized over the exact compressed keys; local tokens and the attention sink are excluded. Output cosine and relative error are computed on the full CSA output, including the unchanged local-attention and sink paths. Relative error is $\|o_{\mathrm{LI}} - o_{\mathrm{exact}}\|_2 / \|o_{\mathrm{exact}}\|_2$.

B	Set recall	Mass	Output cosine	Relative error
512	0.810	0.913	0.9981	5.2%
1024	0.964	0.981	0.9998	1.3%
2048	0.990	0.995	1.0000	0.4%
4096	0.993	0.997	1.0000	0.3%

H.2 Comparison with alternative proposal mechanisms

Protocol and baseline reproduction. We reproduce MISA, MISA†, and HISA from the captured DeepSeek-V3.2 indexer states. All methods use the same final retrieval budget $K = 2048$ and are evaluated against the exact DSA Top-2048 set. The main comparison uses the same $28{,}548$ query–layer instances with visible prefix longer than $16{,}384$, so none of the tested $B \in \{4096, 8192, 16384\}$ settings is trivially saturated.

For MISA, we follow the published content-dependent router: keys are mean-pooled in blocks of 1024, each head is ranked by its average absolute weighted ReLU response over the pooled prefix, and the top-h heads score the full sequence. MISA† retains the top-B entries under this routed score and re-ranks them with the original 64-head DSA indexer. For HISA, we fix the block size to 128 tokens and vary the block budget m to match the nominal proposal budget $B = 128m$. The published default corresponds to $m = 64$ ($B = 8192$). Because the first block and last two blocks are always included, the actual candidate set can be slightly larger than the nominal B; we therefore charge HISA using its realized candidate count B_t^{HISA}.

Compute accounting. We compare methods using only the dominant query–key scoring arithmetic; lightweight preprocessing and reduction costs—including MISA/HISA key aggregation, Linear Indexer query folding, Top-k selection, and other kernel-level overheads—are excluded. For each query at position t, let n_t be the number of visible tokens and h the number of heads routed by MISA. We take one full single-head scan over the n_t visible tokens as one unit of compute and omit the common head dimension d.

The per-query costs before normalization by n_t are

$$
\begin{aligned}
C_t^{\mathrm{MISA}} &= h n_t + 64 \lceil n_t/1024 \rceil , \\
C_t^{\mathrm{MISA}^\dagger} &= C_t^{\mathrm{MISA}} + 64B, \\
C_t^{\mathrm{HISA}} &= 64 \lceil n_t/128 \rceil + 64 B_t^{\mathrm{HISA}}, \\
C_t^{\mathrm{Linear}} &= \begin{cases} n_t, & B = K, \\ n_t + 64B, & B > K. \end{cases}
\end{aligned}
$$

MISA scans the visible sequence with its h routed heads and scores the pooled blocks with all 64 heads; MISA† additionally re-scores its Top-B shortlist with the full indexer. HISA scores all 128-token blocks with 64 heads and then re-scores its selected candidates. The Linear Indexer uses one full linear scan and, for $B > K$, re-scores the Top-B shortlist. We report the mean of C_t/n_t over the same query rows used for IoU evaluation, so one full single-head scan has normalized cost 1.

Full matched comparison. Table 9 gives the complete budget sweep. At the default $B = 8192$, Linear Indexer reaches 0.9535 IoU at 18.53, compared with 0.9269 at 25.59 for MISA†. HISA provides an almost matched-compute comparison at 18.10 but reaches only 0.7510 IoU.

Breakdown across layers. The aggregate advantage is broad rather than concentrated in a few unusual layers. At the default $B = 8192$, Linear Indexer exceeds MISA† on 54 of 61 layers and HISA on 60 of 61 (Fig. 14). The first few layers are difficult for all methods, but once the indexer stabilizes, the linear proposal is usually the strongest of the compared approaches.

Table 9: Matched proposal comparison on DeepSeek-V3.2. All methods use the same 28,548 non-saturated query rows and final budget $K = 2048$.

Method	Setting	Cost	Recall	IoU
Linear Indexer, direct	–	**1.00**	**0.7113**	**0.5643**
MISA	$h = 8$	8.06	0.6561	0.4980
Linear Indexer + refine	$B = 4096$	9.76	**0.8982**	**0.8292**
MISA†	$B = 4096$	16.83	0.8527	0.7557
HISA	$B = 4096$	**9.37**	0.6932	0.5479
Linear Indexer + refine	$B = 8192$	18.53	**0.9739**	**0.9535**
MISA†	$B = 8192$	25.59	0.9595	0.9269
HISA	$B = 8192$	**18.10**	0.8466	0.7510
Linear Indexer + refine	$B = 16384$	36.06	**0.9958**	**0.9921**
MISA†	$B = 16384$	43.12	0.9941	0.9888
HISA	$B = 16384$	**35.54**	0.9512	0.9146

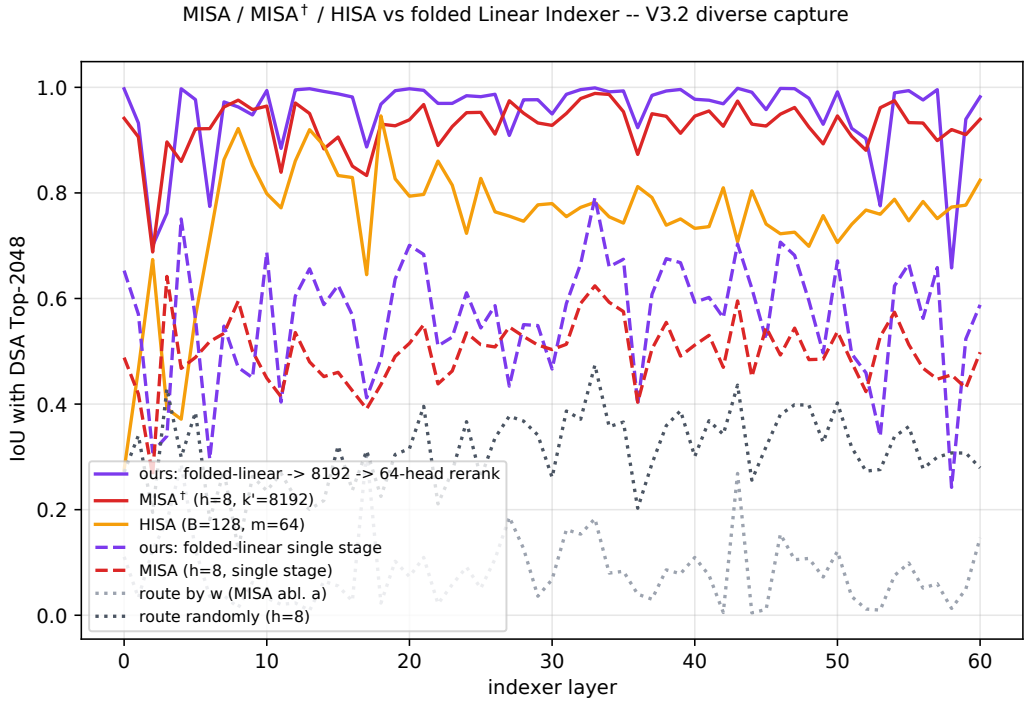


Figure 14: IoU with the exact DSA Top-2048 set, resolved by DeepSeek-V3.2 indexer layer at the default $B = 8192$ setting. The folded Linear Indexer exceeds MISA† on 54 of 61 layers and HISA on 60 of 61, showing that the aggregate advantage is broad rather than driven by a small number of unusual layers. Dashed and dotted curves show the corresponding single-stage variants and MISA router controls.

Indexer latency on 64K capture. Table 10 complements the hardware-independent compute accounting with measured latency. We base MISA and HISA on their official released implementations[1] and integrate their kernels into a common evaluation path. All methods receive the same prepared post-RoPE FP8 inputs and are timed through final Top-2048 selection. We construct a fixed 8,192-row timing layout from query rows in the same 64K capture and report median CUDA-event latency after 10 warm-up iterations and over 20 timing trials. Each speedup uses exact DSA measured in the same isolated worker.

[1] https://github.com/MuLabPKU/TransArch

Table 10: Index-selection latency on a 64K DeepSeek-V3.2 capture. Timing includes final Top-2048 selection, and speedup is relative to exact DSA measured in the corresponding isolated worker.

Method	Setting	Latency (ms)	Speedup
Exact DSA	–	18.26	1.00×
Linear Indexer, direct	$B = 2048$	**2.51**	**7.26**×
MISA	$h = 8$	10.14	1.80×
Linear Indexer + refine	$B = 4096$	15.19	1.20×
MISA†	$B = 4096$	22.55	0.81×
HISA	$B = 4096$	7.20	2.54×
Linear Indexer + refine	$B = 8192$	18.57	0.98×
MISA†	$B = 8192$	25.84	0.71×
HISA	$B = 8192$	11.45	1.60×
Linear Indexer + refine	$B = 16384$	24.99	0.73×
MISA†	$B = 16384$	32.28	0.57×
HISA	$B = 16384$	20.90	0.87×

The direct Linear Indexer is fastest at 2.51 ms (7.26×), while refinement with $B = 4096$ remains faster than exact DSA at 15.19 ms (1.20×). At larger proposal budgets, selection and reranking overhead can outweigh the reduction in scoring arithmetic.

H.3 Detailed indexer and prefill efficiency results

Indexer latency scaling. Following the measurement protocol in Appendix G.4, Figure 15 shows the complete 8K–256K indexer-level sweep. The Linear Indexer becomes increasingly advantageous as context length grows, while smaller candidate budgets move the crossover to shorter sequences. At 256K, speedup reaches 6.0×, 7.4×, and 10.4× for $B = 2048$, 1024, and 512, respectively. At short contexts, refinement and selection overhead can outweigh the cheaper global scan, motivating the budget-dependent dispatch in Appendix G.3.

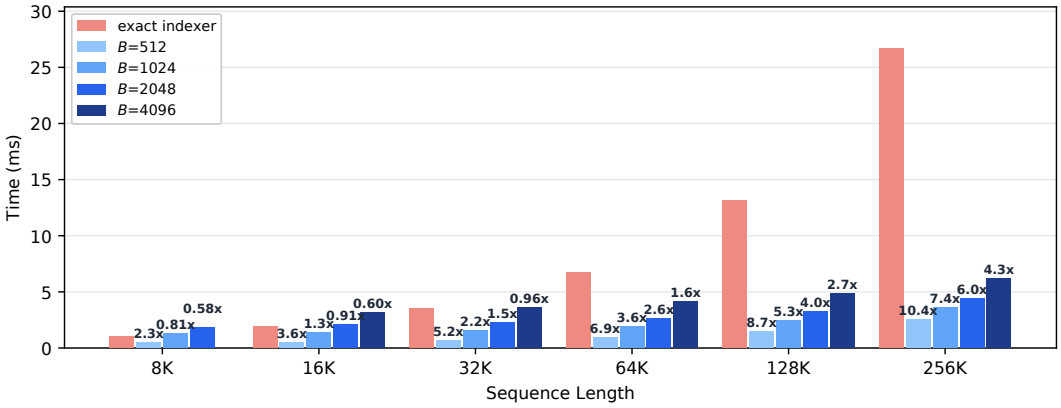


Figure 15: Single-layer indexer latency on one H20 across context lengths. Labels report speedup over the exact nonlinear indexer. The short-context crossover motivates the budget-dependent dispatch in Appendix G.3.

Served prefill. Table 11 reports absolute single-request TTFT and speedup over exact CSA. Even at 16K and 32K, direct $B = 512$ begins to translate its kernel-level savings into end-to-end prefill gains, yielding 2.5% and 5.4% TTFT speedups, respectively. The gains increase from 64K onward. At 256K, exact CSA requires 36.52 s, compared with 27.39 s, 26.85 s, and 26.25 s for $B = 2048$, 1024, and 512, corresponding to 1.33×, 1.36×, and 1.39× speedups.

Table 11: Single-request prefill TTFT on 8×H20 with TP8 and speedup over exact CSA. Each point is the median of three timing trials; speedups are computed from unrounded TTFT. All Linear Indexer arms use the budget-dependent dispatch in Appendix G.3.

Method	16K	32K	64K	128K	256K
TTFT (s; lower is better)					
Exact CSA	1.323	2.770	6.050	14.131	36.517
Linear, $B = 2048$	1.323	2.734	5.706	12.192	27.391
Linear, $B = 1024$	1.318	2.694	5.587	11.931	26.853
Linear, $B = 512$	1.291	2.627	5.444	11.624	26.249
Speedup over exact CSA (×; higher is better)					
Linear, $B = 2048$	0.999	1.013	1.060	1.159	1.333
Linear, $B = 1024$	1.003	1.028	1.083	1.184	1.360
Linear, $B = 512$	1.025	1.054	1.111	1.216	1.391

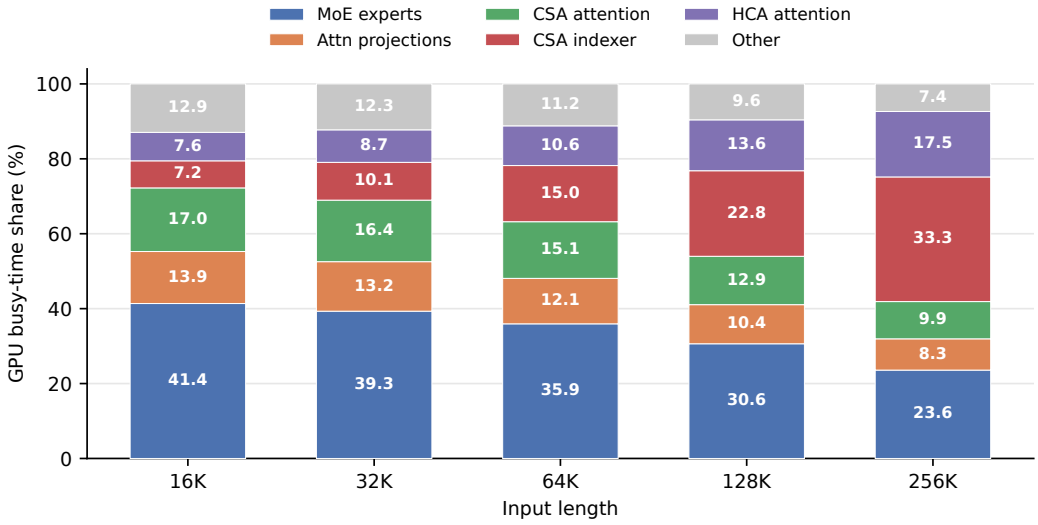


Figure 16: Batch-size-1 breakdown of prefill GPU busy time for exact CSA on 8×H20 with TP8. The CSA indexer becomes increasingly prominent as context grows, accounting for 33.3% of measured GPU busy time at 256K.

Prefill-time composition. Figure 16 profiles the prefill GPU busy time of exact CSA at batch size 1. The indexer share grows monotonically from 7.2% at 16K to 15.0% at 64K and 33.3% at 256K, where it becomes the largest individual component. This trend explains why TTFT gains increase with context length, while the unmodified components limit the end-to-end speedup.

H.4 Downstream Quality on Long-Context Benchmarks

We evaluate whether the approximation introduced by the Linear Indexer affects end-task quality on RULER and LongBench-v2. We compare the exact CSA indexer with three candidate budgets, $B \in \{512, 1024, 2048\}$. All methods use the same evaluation instances and decoding configuration within each benchmark. The Linear Indexer uses the budget-dependent dispatch thresholds in Table 6, with $B = 512$ operating in direct linear mode. Confidence intervals are obtained by paired bootstrap over evaluation examples.

For RULER, we evaluate all methods on the same 50 examples for each of the 13 RULER tasks at every context length, yielding 650 paired examples per length. For `ruler_vt`, we increase the maximum generation length from 30 to 128 tokens to prevent correct variable lists from being truncated; all other task configurations remain unchanged. For LongBench-v2, we evaluate 407 of the 503 examples whose tokenized inputs fit within the 256K context limit.

Table 12: Downstream quality under different candidate budgets. RULER reports the mean score over 13 tasks; "Mean" averages all four context length settings. LongBench-v2 reports accuracy on 407 shared examples. The lower block gives paired differences from exact CSA in percentage points, with 95% bootstrap confidence intervals.

	RULER					LongBench-v2
Method	32K	64K	128K	256K	Mean	Acc.
Exact CSA	0.954	0.938	0.948	0.942	0.945	0.484
Linear Indexer, $B = 2048$	0.952	0.934	0.948	0.944	0.944	0.499
Linear Indexer, $B = 1024$	0.959	0.943	0.951	0.944	0.949	0.484
Linear Indexer, $B = 512$	0.956	0.939	0.944	0.943	0.945	0.484
Difference from exact CSA in percentage points (paired 95% CI)						
Linear Indexer, $B = 2048$	-0.11% $[-0.42\%, +0.19\%]$					$+1.5\%$ $[-1.1\%, +4.1\%]$
Linear Indexer, $B = 1024$	$+0.36\%$ $[+0.01\%, +0.73\%]$					0.0% $[-2.4\%, +2.4\%]$
Linear Indexer, $B = 512$	-0.01% $[-0.38\%, +0.36\%]$					0.0% $[-2.7\%, +2.6\%]$

RULER. As shown in Table 12, all three Linear Indexer configurations closely track exact CSA across the full context range. Their mean scores lie between 0.944 and 0.949, compared with 0.945 for exact CSA. The corresponding differences are -0.11%, $+0.36\%$, and -0.01% for $B = 2048$, 1024, and 512, respectively.

The task-level results in Table 13 show that this stability is broad rather than an artifact of averaging. The eight needle-retrieval tasks almost remain between 0.98 and 1.00. Variable tracking is likewise nearly saturated, with scores between 0.98 and 1.00 throughout. The remaining tasks exhibit small, bidirectional fluctuations: no candidate budget consistently decreases `ruler_cwe`, `ruler_fwe`, or either question-answering score across context lengths.

The small positive mean shift at $B = 1024$ is therefore not reproduced at adjacent budgets or associated with a consistent task family. We do not interpret it as evidence that reducing the candidate set improves quality. Instead, the aggregate and task-level results jointly show no systematic downstream degradation for candidate budgets as small as $B = 512$.

LongBench-v2. LongBench-v2 leads to the same conclusion. Exact CSA obtains an aggregate accuracy of 0.484, while the three Linear Indexer configurations obtain 0.499, 0.484, and 0.484 for $B = 2048$, 1024, and 512, respectively. All paired confidence intervals include zero. The positive point estimate for $B = 2048$ is therefore not statistically distinguishable from the exact indexer.

Taken together with the RULER results, these evaluations show that the budget-dependent reduction in candidate-level retrieval fidelity does not translate into a detectable loss in downstream quality.

Table 13: Per-task RULER scores at 32K–256K. **E** denotes exact CSA, and the remaining columns denote the candidate budget B of the Linear Indexer. Each cell is averaged over 50 matched examples. The mean rows average all 13 tasks and reproduce the RULER results in Table 12.

(a) 32K and 64K

	32K				64K			
Task	**E**	2048	1024	512	**E**	2048	1024	512
niah_single_1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
niah_single_2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
niah_single_3	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
niah_multikey_1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
niah_multikey_2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
niah_multikey_3	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
niah_multiquery	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
niah_multivalue	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
ruler_vt	1.00	1.00	1.00	1.00	0.99	0.98	1.00	1.00
ruler_cwe	0.93	0.93	0.91	0.94	0.93	0.91	0.92	0.92
ruler_fwe	0.99	0.99	0.99	0.99	0.89	0.89	0.89	0.91
ruler_qa_squad	0.78	0.78	0.80	0.80	0.69	0.70	0.72	0.68
ruler_qa_hotpot	0.70	0.68	0.76	0.70	0.70	0.66	0.72	0.70
Mean	0.954	0.952	0.959	0.956	0.938	0.934	0.943	0.939

(b) 128K and 256K

	128K				256K			
Task	**E**	2048	1024	512	**E**	2048	1024	512
niah_single_1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
niah_single_2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
niah_single_3	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
niah_multikey_1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
niah_multikey_2	0.98	0.98	0.98	0.98	1.00	1.00	1.00	1.00
niah_multikey_3	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.96
niah_multiquery	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
niah_multivalue	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
ruler_vt	1.00	1.00	0.99	0.99	1.00	1.00	1.00	1.00
ruler_cwe	0.92	0.94	0.95	0.92	0.95	0.96	0.96	0.96
ruler_fwe	1.00	1.00	1.00	1.00	0.90	0.91	0.91	0.92
ruler_qa_squad	0.71	0.72	0.72	0.70	0.71	0.72	0.72	0.72
ruler_qa_hotpot	0.71	0.68	0.72	0.68	0.68	0.68	0.68	0.70
Mean	0.948	0.948	0.951	0.944	0.942	0.944	0.944	0.943