RESMISA: RESIDUAL MIXTURE OF SPARSE ATTENTION FOR LONG-CONTEXT LLM INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-head token scoring in the DeepSeek Sparse Attention (DSA) indexer remains costly for long-context inference. MISA reduces this cost by activating a subset of indexer heads as scoring experts, but omits the remaining heads' contributions and their direct gradient paths through the indexing score during training. Inspired by residual connections and shared experts, we introduce ResMiSA, which uses $\mathrm{ReLU}(x) = x + \mathrm{ReLU}(-x)$ to decompose each head's response into an identity branch and a nonlinear residual branch. Because all indexer heads share the same key, their weighted identity branches merge exactly into one always-active shared expert that requires a single dot product per key. Routed residual experts supply nonlinear corrections, while the shared path retains a direct gradient path through every head. Applied without additional training to DeepSeek-V3.2 across 4K–128K contexts, one shared expert and seven residual experts replace 64-head scoring, with RULER scores only 0.06 percentage points below DSA on average and 0.21 points below at 128K. In frozen-backbone indexer warmup, ResMiSA achieves slightly lower attention-fitting loss and higher attention mass recall on training batches than DSA, using only $1/4$ or $1/8$ as many per-key scoring channels at the same trainable parameter count and training steps. Combined with LiteTopK, ResMiSA achieves a $2.43\times$ end-to-end time-to-first-token (TTFT) speedup over DSA at 1M context.

1 INTRODUCTION

Long-context language models must locate relevant evidence within a growing history while keeping retrieval cost manageable. DeepSeek Sparse Attention (DSA) addresses this problem with a lightweight multi-head indexer that selects tokens for subsequent sparse attention (Liu et al., 2025). Selecting a small token set, however, still requires scoring the accessible history and performing token top-K selection. In long-context prefill, repeating this scan for many queries makes the cost of discovering sparsity an important optimization target in its own right.

Existing indexer optimizations reduce repeated work across layers, keys, queries, or heads (Bai et al., 2026; Xu et al., 2026; Liu et al., 2026; Zhou et al., 2026). Head sparsification is particularly attractive because it reduces the dot products used to score each token while retaining access to the legal history. MISA views indexer heads as scoring experts and routes only a small subset for each query (Zhou et al., 2026). Yet removing a head discards its entire score contribution. When the same head selection is used during training, unselected heads also lose their direct gradient paths through the indexing score. This motivates a shared scoring path: *can we retain an inexpensive contribution and a direct gradient path from every head, while routing only nonlinear corrections?*

Inspired by residual connections and shared experts, we introduce ResMiSA, a score-level residual design built around an analytical shared expert (Figure 1). Using $\mathrm{ReLU}(x) = x + \mathrm{ReLU}(-x)$, we decompose each head's response into an identity branch and a nonlinear residual branch. Because all indexer heads share the same key, their weighted identity branches can be exactly merged by summing the weighted query vectors into a single query-dependent vector. This yields an always-active linear shared expert that requires just one dot product per key and introduces no new parameters. Routed residual experts apply ReLU to negated head responses to supply nonlinear corrections, while the shared path preserves every head's linear contribution and a direct gradient path even when its residual is not selected.

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.

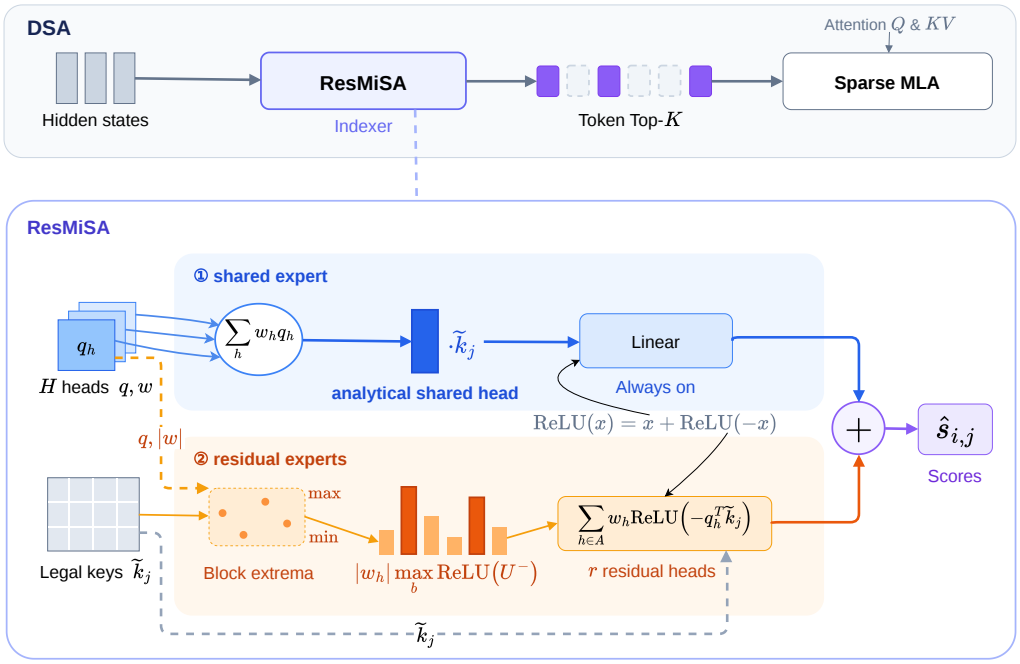


Figure 1: Overview of ResMiSA within the DSA pipeline. All indexer heads contribute to an always-active linear shared expert. Upper bounds over legal keys select residual experts whose nonlinear corrections refine the shared score. The resulting scores drive token top-K selection for Sparse MLA.

To select residual experts, we derive upper bounds on their contribution magnitudes from coordinate-wise extrema of legal keys. Unlike estimates based on block-mean keys, these bounds account for strong individual-token residuals that averaging can obscure. To translate the reduced scoring computation into inference speedups, we jointly evaluate the shared and routed paths in CUDA and integrate LiteTopK for fused score generation and candidate filtering, followed by final top-K selection.

Contributions. (i) We formulate sparse indexing as an all-head shared contribution plus routed residual corrections, preserving a direct scoring path through every head without introducing new parameters. (ii) We derive a routing rule based on legal-key extrema that minimizes a separable upper bound on omitted residuals at a fixed budget, and implement shared–residual scoring with LiteTopK candidate filtering. (iii) We evaluate quality, attention fitting, and latency: eight scoring channels replace 64 on DeepSeek-V3.2 with a 0.06-percentage-point decrease in average RULER score over 4K–128K; frozen-backbone indexer warmup yields slightly lower fitting loss with four or eight channels at equal parameter counts and training steps; and the implementation achieves $2.43\times$ and $2.19\times$ TTFT speedups over DSA and LiteTopK alone at 1M.

2 Related Work

Sparse attention and content-dependent retrieval. Long-context inference can be accelerated by restricting attention connectivity, as in BigBird (Zaheer et al., 2020), or compressing the retained KV history, as in SnapKV (Li et al., 2024). Static patterns such as factorized strided attention (Child et al., 2019) and sliding windows with global tokens (Beltagy et al., 2020) are input-independent, while sink- or heavy-hitter-based eviction (Xiao et al., 2024; Zhang et al., 2023) discards keys that later queries cannot recover. Content-dependent methods adapt the retrieved context to each query: MoBA routes queries to KV blocks, while NSA combines compression with fine-grained selection (Lu et al., 2025; Yuan et al., 2025). Quest uses coordinate-wise key extrema to identify relevant KV pages (Tang et al., 2024). ResMiSA uses the same type of statistics for a different decision: bounding head-wise residual contributions to allocate a sparse scoring budget.

Head sparsity and shared experts. Conditional head computation activates a subset of heads for each token. MoA routes attention heads, while MoH combines always-active shared heads with token-dependent routed heads (Zhang et al., 2022; Jin et al., 2024). MISA applies this principle to the DSA indexer by selecting complete positive-ReLU scoring experts; its hierarchical variant then reranks an expanded candidate set using all heads (Zhou et al., 2026). Our shared expert is constructed analytically from every indexer head's linear contribution, and routing acts only on the remaining nonlinear corrections. This construction preserves an all-head scoring path without adding trainable parameters or reserving a subset of the original heads as shared experts.

Improvements to DSA. Recent work reduces indexer cost along several dimensions. IndexCache reuses token indices across layers (Bai et al., 2026); HISA performs coarse-to-fine KV selection (Xu et al., 2026); PIVOT reduces query-side work using proxy queries (Liu et al., 2026); and SparDA overlaps CPU-to-GPU prefetching with computation (Fu et al., 2026). MISA reduces the number of scoring heads (Zhou et al., 2026). ResMiSA also targets head-wise computation, but changes the unit of sparsification from an entire head response to its nonlinear residual.

Top-K and indexer kernels. Reducing the cost of score generation complements efficient token selection. GVR exploits temporal correlation for exact guess–verify–refine top-K selection during decoding (Cheng et al., 2026). LiteTopK fuses indexer scoring with candidate filtering to reduce intermediate-score storage and selection traffic (Yin et al., 2026). We integrate ResMiSA with LiteTopK: a fused CUDA kernel evaluates the shared and residual paths and filters candidates, followed by a separate final top-K stage.

3 PRELIMINARIES

Legal candidates and indexer representations. DeepSeek Sparse Attention (DSA) (Liu et al., 2025) separates token retrieval from attention computation: an indexer selects candidate tokens, and Sparse MLA computes attention over the selected set. An indexing call contains M queries and N cached keys; query i can access only token indices in $I_i = [s_i, e_i)$, with $0 \le s_i \le e_i \le N$. The interval encodes the keys accessible to a query, including causal constraints, chunked prefill, and request-specific cache boundaries. The indexer has H query heads of dimension D, with query $q_{i,h} \in \mathbb{R}^D$ and effective weight $w_{i,h} \in \mathbb{R}$. All indexer heads share the indexing key $\tilde{k}_j \in \mathbb{R}^D$ for token j; these representations are distinct from the downstream attention queries, keys, and values.

Full-head indexing. DSA aggregates weighted positive-ReLU responses to score each legal token:

$$s_{i,j}^{\text{full}} = \sum_{h=1}^{H} w_{i,h} \,\mathrm{ReLU}(q_{i,h}^{\top} \tilde{k}_j), \qquad j \in I_i. \tag{1}$$

Our analysis uses the numerical values of queries and keys and is independent of their storage and accumulation formats. If a representation uses positive dequantization scales, the query scale can be absorbed into $w_{i,h}$ by positive homogeneity of ReLU, while $\tilde{k}_j$ includes the per-token key scale.

Scoring experts and MISA. We refer to each weighted head response in Equation 1 as a scoring expert. MISA estimates head importance using block-mean keys (Zhou et al., 2026). Let $\mathcal{B}_i$ denote the blocks of the prefix considered by its router, and let m be its active-head budget:

$$\begin{aligned} \bar{k}_b &= \frac{1}{|b|} \sum_{j \in b} \tilde{k}_j, \\ E_{i,h}^{\text{MISA}} &= \frac{1}{|\mathcal{B}_i|} \sum_{b \in \mathcal{B}_i} |w_{i,h}| \,\mathrm{ReLU}(q_{i,h}^{\top} \bar{k}_b). \end{aligned} \tag{2}$$

It selects the m highest-scoring heads and uses only these experts for token scoring:

$$\begin{aligned} A_i^{\text{MISA}} &= \mathrm{TopK}_h(E_{i,h}^{\text{MISA}}, m), \\ \hat{s}_{i,j}^{\text{MISA}} &= \sum_{h \in A_i^{\text{MISA}}} w_{i,h} \,\mathrm{ReLU}(q_{i,h}^{\top} \tilde{k}_j). \end{aligned} \tag{3}$$

This selection removes the entire response of each unselected head. ResMiSA instead decomposes these responses before routing, retaining every head's linear contribution in a shared expert (Section 4). Throughout this paper, an expert denotes a scalar component of the indexing score; the downstream attention heads remain unchanged.

Token selection and the Sparse MLA interface. For token budget K, let $K_i = \min(K, |I_i|)$ and define

$$\mathcal{T}_i^{\mathrm{DSA}} = \mathrm{TopK}_{j \in I_i}(s_{i,j}^{\mathrm{full}}, K_i). \tag{4}$$

An empty legal interval yields an empty set. Given a nonempty selected set $\mathcal{T}_i$, the semantics of sparse attention for attention head a are

$$o_{i,a} = \sum_{j \in \mathcal{T}_i} \frac{\exp(\ell_{i,a,j})}{\sum_{t \in \mathcal{T}_i} \exp(\ell_{i,a,t})} v_{j,a}, \tag{5}$$

where $\ell_{i,a,j}$ is the model's actual attention logit, including its required positional terms and scaling, and $v_{j,a}$ is its corresponding value. ResMiSA modifies the indexer that constructs $\mathcal{T}_i$ while retaining this Sparse MLA attention rule.

4 METHOD

ResMiSA separates each head's response into a linear contribution that is always retained and a nonlinear correction that can be routed sparsely. The linear contributions of all H heads collapse into a single query-dependent shared expert; r selected residual experts then refine its score, with $0 \le r \le H$. The resulting scores determine the top-K_i tokens passed to Sparse MLA.

4.1 RESIDUAL SCORING WITH AN ALL-HEAD ANALYTICAL SHARED EXPERT

Dropping an original scoring expert removes both its linear contribution and its nonlinear correction. Instead, we use $\mathrm{ReLU}(x) = x + \mathrm{ReLU}(-x)$ to rewrite Equation 1 as

$$\begin{aligned} s_{i,j}^{\mathrm{full}} &= \sum_{h=1}^{H} w_{i,h} \,\mathrm{ReLU}(q_{i,h}^{\top} \tilde{k}_j) \\ &= \sum_{h=1}^{H} w_{i,h} q_{i,h}^{\top} \tilde{k}_j + \sum_{h=1}^{H} w_{i,h} \,\mathrm{ReLU}(-q_{i,h}^{\top} \tilde{k}_j) \\ &= \left(\sum_{h=1}^{H} w_{i,h} q_{i,h} \right)^{\top} \tilde{k}_j + \sum_{h=1}^{H} w_{i,h} \,\mathrm{ReLU}(-q_{i,h}^{\top} \tilde{k}_j), \\ q_i^{\mathrm{share}} &= \sum_{h=1}^{H} w_{i,h} q_{i,h}. \end{aligned} \tag{6}$$

The vector q_i^{share} defines an always-active **shared expert**. Because every head uses the same key, their weighted query vectors can be summed once per query, reducing the entire linear branch to one dot product per key. This analytical construction introduces no new parameters and preserves every head's linear contribution regardless of which residuals are selected.

Evaluating all H residuals would retain the full score but offer no reduction in per-key dot products. We therefore select a query-dependent set A_i of r **residual experts**, each applying ReLU to the negated head response. The approximate score is

$$\hat{s}_{i,j} = (q_i^{\mathrm{share}})^{\top} \tilde{k}_j + \sum_{h \in A_i} w_{i,h} \,\mathrm{ReLU}(-q_{i,h}^{\top} \tilde{k}_j). \tag{7}$$

Final per-key scoring requires $1 + r$ dot products, fewer than full-head scoring when $1 + r < H$. Only unselected nonlinear corrections are omitted; every head retains its shared contribution.

Figure 2 illustrates how the two paths reconstruct DSA scores. Under the entrywise similarity metric in Appendix A, the shared expert alone reaches 64.85% mean similarity; adding seven residual experts raises it to 80.52%. Similarity improves with the residual budget, with diminishing gains per additional scoring channel across the tested configurations. We use $1 + 7$ to balance fidelity and computation while matching MISA's eight-channel final scoring budget. The higher measured zero rates of negative-ReLU responses further motivate sparsifying this branch (Section 5.4); Section 5 evaluates selection quality.

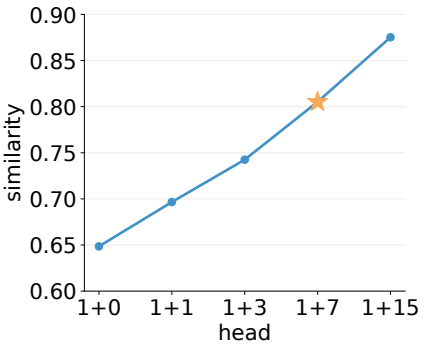


Figure 2: Mean DSA score similarity across four datasets. $1 + r$ denotes one shared expert and r residual experts. Details are in Appendix A.

Gradient paths through the shared expert. The exact decomposition preserves the original ReLU derivative away from zero; truncating the residual branch changes this local gradient structure. With the routing set held fixed, let $x = q_{i,h}^{\top}\tilde{k}_j$. For a selected residual expert, the shared and residual derivatives combine to recover $w_{i,h}\mathbf{1}[x > 0]$. For an unselected residual expert, the shared term retains derivative $w_{i,h}$, including when $x < 0$. Thus, leaving a residual unselected does not remove its head's direct path through the indexing score, although the resulting gradient differs from full DSA. Section 5.6 evaluates attention fitting under this shared–residual parameterization.

4.2 Legal-Interval Upper-Bound Routing

The shared expert captures the full linear branch, so routing should prioritize the nonlinear corrections whose omission could most affect token scores. Block-mean estimates can conceal large residuals on individual tokens. We instead use coordinate-wise key extrema to bound each head's residual magnitude over the legal interval. We partition the keys into blocks of size B and, for every nonempty intersection $J_{i,b} = b \cap I_i$, form coordinate-wise key extrema:

$$k_{i,b,d}^{\min} = \min_{j \in J_{i,b}} \tilde{k}_{j,d}, \qquad k_{i,b,d}^{\max} = \max_{j \in J_{i,b}} \tilde{k}_{j,d}. \tag{8}$$

Only keys in I_i enter the statistics, so inaccessible tokens cannot affect a query's residual routing.

The resulting coordinate-wise box bounds the negative response of every head on every legal key. With $q_{i,h}^{+} = \max(q_{i,h}, 0)$ and $q_{i,h}^{-} = \max(-q_{i,h}, 0)$ elementwise, any $j \in J_{i,b}$ satisfies

$$U_{i,b,h}^{-} = (q_{i,h}^{-})^{\top} k_{i,b}^{\max} - (q_{i,h}^{+})^{\top} k_{i,b}^{\min} \geq -q_{i,h}^{\top}\tilde{k}_j. \tag{9}$$

ReLU monotonicity then bounds the correction magnitude by $|w_{i,h}|\,\mathrm{ReLU}(U_{i,b,h}^{-})$. For each head, we take the largest bound across accessible blocks and select the r heads with the highest resulting routing scores:

$$S_{i,h} = |w_{i,h}| \max_{b: J_{i,b} \neq \varnothing} \mathrm{ReLU}(U_{i,b,h}^{-}), \qquad A_i = \mathrm{TopR}_h(S_{i,h}, r). \tag{10}$$

Each query uses one residual-head set for all its legal keys. Absolute weights are used only for routing magnitudes; Equation 7 retains their signs.

This routing rule minimizes a separable upper bound on the omitted residuals at a fixed budget. For every legal token, the shared terms cancel, and the triangle inequality gives

$$\begin{aligned} |s_{i,j}^{\mathrm{full}} - \hat{s}_{i,j}| &= \left| \sum_{h \notin A_i} w_{i,h}\,\mathrm{ReLU}(-q_{i,h}^{\top}\tilde{k}_j) \right| \\ &\leq \sum_{h \notin A_i} |w_{i,h}|\,\mathrm{ReLU}(-q_{i,h}^{\top}\tilde{k}_j) \leq \sum_{h \notin A_i} S_{i,h}. \end{aligned} \tag{11}$$

Since the sum of all $S_{i,h}$ is fixed for a query, selecting its largest r entries minimizes the right-hand side. The guarantee applies to the separable upper bound $\sum_{h \notin A_i} S_{i,h}$; it does not imply optimal

realized score error, token rankings, or downstream accuracy. Appendix D.1 evaluates the routing rule empirically.

To isolate the routing criterion, consider block-mean routing for the same negative residuals, rather than MISA's complete positive-ReLU experts. For a scalar head with $q = w = 1$ and a legal block containing keys $-a$ and a ($a > 0$), the mean key is zero, so its negative-ReLU response is zero, whereas the largest token residual and our extrema bound both equal a. Thus, a block-mean response cannot in general bound a local residual, even as a grows arbitrarily large. The ablations in Appendix D.1 further show that upper-bound routing outperforms mean pooling in attention mass recall at longer context positions and in RULER scores.

4.3 Joint CUDA Scoring and LiteTopK Fusion

Our CUDA kernel jointly evaluates the shared linear term and the selected signed residuals in Equation 7 while scanning legal keys. It incorporates LiteTopK candidate filtering and threshold updates in the same scan, avoiding materialization of a dense intermediate score tensor. A separate final top-K stage selects from the retained candidates and passes their indices to the unchanged Sparse MLA interface. Both routing and selection respect each query's legal interval. Appendix B describes the algorithm, and Section 5.5 measures the combined end-to-end benefit of ResMiSA and LiteTopK.

5 Experiments

We evaluate training-free indexer replacement through RULER, attention mass recall, residual ablations, and end-to-end TTFT on DeepSeek-V3.2. A separate frozen-backbone experiment tests attention fitting during indexer warmup on dots-3-densemla (30B-A3B).

5.1 Experimental Setup

Model and tasks. We evaluate DeepSeek-V3.2 (Liu et al., 2025) on all 13 RULER subtasks (Hsieh et al., 2024), following the HELMET evaluation settings and prompts (Yen et al., 2025). We use 100 examples per subtask at each of six context lengths: 4K, 8K, 16K, 32K, 64K, and 128K. For attention mass recall, we use ten examples sampled from SWE-bench (Jimenez et al., 2024) and cover all 61 layers, reporting results by context position and layer.

Default hyperparameters. Unless otherwise specified, ResMiSA uses one shared expert and seven residual experts in both prefill and decode. The resulting eight per-key scoring channels match MISA's final scoring budget and use one eighth as many channels as the 64-head DSA indexer. This count describes final score evaluation; shared-vector construction and routing incur additional work. We select residual experts by upper-bound routing with $B = 1024$ tokens per block.

Baselines. DSA uses all 64 indexing heads (Liu et al., 2025). IndexCache$_{\frac{1}{2}}$ uses the uniform strategy to retain the indexer in one out of every two consecutive layers; the subscript denotes the fraction of layers that retain indexing (Bai et al., 2026). HISA uses blocks of 128 tokens to select 8192 candidates before full-head filtering (Xu et al., 2026). MISA uses blocks of 1024 tokens and eight heads for final scoring (Zhou et al., 2026). Its two-stage variant, MISA†, expands the first-stage candidate set to 8192 tokens and then uses all heads to select the final 2048 tokens.

5.2 Long-Context Task Quality

Results. Table 1 shows that ResMiSA closely matches full-head DSA with eight per-key scoring channels: the average gap is 0.06 percentage points, and the gap at 128K is 0.21 points. At the same final scoring budget, it exceeds MISA by 1.83 points on average. It also exceeds MISA† by 0.62 points without an additional full-head reranking stage and achieves the highest 128K score among the compared accelerated methods.

Table 1: RULER scores (↑) on DeepSeek-V3.2. All sparse methods are applied at inference time without additional training. **Bold** and underlined values denote the best and second-best scores in each column.

Method	4K	8K	16K	32K	64K	128K	Avg.
DSA	**96.56**	96.19	96.17	**95.99**	**95.15**	**93.65**	**95.62**
IndexCache $\frac{1}{2}$	96.49	96.16	**96.21**	95.69	94.71	89.80	94.84
HISA	96.41	96.21	94.97	83.96	67.92	55.79	82.54
MISA	96.15	96.15	96.14	95.52	92.95	85.44	93.73
MISA†	96.20	96.25	96.00	95.74	94.27	91.19	94.94
ResMiSA	96.53	**96.39**	96.06	95.98	94.96	93.44	95.56

5.3 Attention Mass Recall

Metric. To examine token selection directly, we measure the dense-attention probability mass retained by each indexer, normalized by the maximum mass attainable at the same token budget. Let I_i be the set of legal keys for query i, and let $\ell_{i,a,j}$ denote its full dense-attention logit for downstream attention head a and key $j \in I_i$, including content and positional terms with the appropriate scale. We first apply softmax over legal keys within each attention head, then average the resulting probabilities over the H_{attn} attention heads:

$$p_{i,j} = \frac{1}{H_{\text{attn}}} \sum_{a=1}^{H_{\text{attn}}} \frac{\exp(\ell_{i,a,j})}{\sum_{t \in I_i} \exp(\ell_{i,a,t})}. \tag{12}$$

These are downstream attention heads, not indexer heads. We use a token budget of $K = 2048$, with the effective budget limited by the number of legal keys for each query:

$$K_i = \min(K, |I_i|), \qquad \mathcal{O}_i = \mathrm{TopK}_{j \in I_i}(p_{i,j}, K_i). \tag{13}$$

For a method's legal, deduplicated selection $\mathcal{S}_i$ of size K_i, attention mass recall is

$$\mathrm{Recall}_i = \frac{\sum_{j \in \mathcal{S}_i} p_{i,j}}{\sum_{j \in \mathcal{O}_i} p_{i,j}}. \tag{14}$$

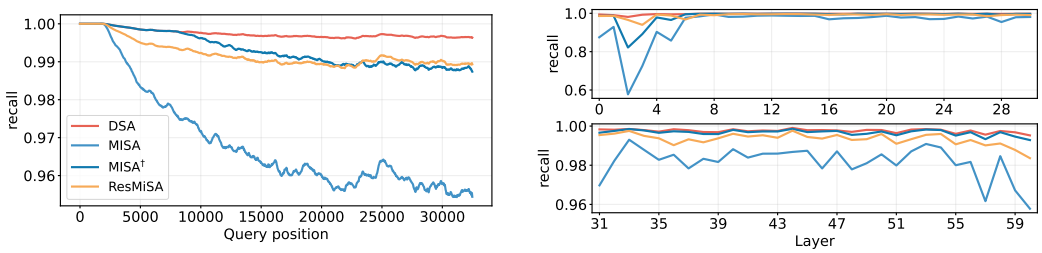
The reference $\mathcal{O}_i$ maximizes retained probability mass at the same token budget; it equals the full legal history only when $|I_i| \le K$. For nonempty I_i, this ratio lies in $[0, 1]$ and counts the probability mass of all selected tokens. Weighting tokens by their dense-attention probabilities distinguishes retaining high-mass evidence from merely matching token identities, as measured by set overlap.

Results. Figure 3 reports recall by context position and layer. ResMiSA retains more attention mass than MISA at the same final scoring budget and surpasses MISA† beyond approximately 23K in the displayed position curve. It ranks second to DSA in overall recall among the compared indexers and retains approximately 99% of the same-budget reference mass near 30K.

5.4 Positive- and Negative-ReLU Zero Rates

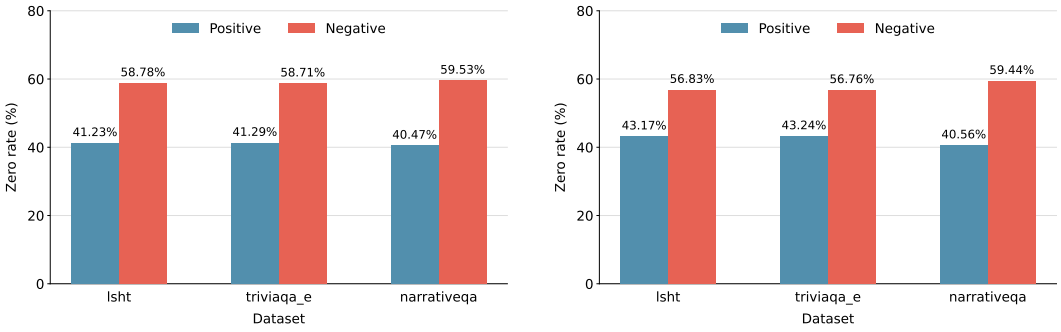
To characterize the branch being sparsified, we compare zero rates for $\mathrm{ReLU}(x)$ and $\mathrm{ReLU}(-x)$, where $x = q_{i,h}^\top \tilde{k}_j$ is a scalar head response before weighting and summation. We count all legal query–key positions, excluding masked entries, and pool counts across layers. The measurement therefore covers the full legal score domain, including tokens outside the selected top-K set. Appendix C provides detailed settings.

Results. Figure 4 shows higher negative-ReLU zero rates in all six model–dataset settings: 58.71–59.53% versus 40.47–41.29% on DeepSeek-V3.2, and 56.76–59.44% versus 40.56–43.24% on GLM-5. The negative-ReLU branch is therefore active at fewer measured scalar positions, supporting its use as the sparsified correction to a dense shared contribution. Scalar zero rates motivate this choice; the ablations evaluate head-level omission.



(a) By context position. (b) By layer.

Figure 3: Attention mass recall on ten SWE-bench examples across 61 layers. The compared indexers are DSA, MISA, MISA†, and ResMiSA; higher values indicate better retention relative to the same-budget dense-attention top-K reference.



(a) DeepSeek-V3.2 (b) GLM-5

Figure 4: Positive- and negative-ReLU zero rates using ten examples per dataset. DeepSeek-V3.2 includes 61 layers and 64 heads per layer; GLM-5 includes 78 layers and 32 heads per layer.

Ablation studies. Appendix D presents three ablation studies on the residual-routing criterion (upper-bound versus mean-pooling routing), the residual-head budget, and dynamic versus fixed residual-head selection.

5.5 End-to-End Time to First Token

We compare three configurations to distinguish the end-to-end benefit from that of LiteTopK alone: DSA without LiteTopK, the full DSA indexer with LiteTopK, and ResMiSA with LiteTopK. All use vLLM with tensor parallelism of eight, expert parallelism enabled, maximum concurrency four, and a prefill chunk size of 8192 tokens. We perform ten warm-up runs followed by ten measured requests, with each request generating one output token, and report mean TTFT in seconds. Maximum concurrency is a scheduling limit, not a fixed batch size.

Results. ResMiSA+LiteTopK reduces TTFT at every measured context length, with the largest gains at 1M: $2.43\times$ over DSA and $2.19\times$ over LiteTopK alone (Table 2). The comparison with LiteTopK isolates the incremental end-to-end benefit of adding shared–residual scoring to that pipeline, including its interaction with candidate filtering and final selection. These latency results extend to 1M context; the separate task-quality evaluation covers 4K–128K. Appendices E and F report concurrency sweeps from 1 to 16 and profiled indexer component times, respectively.

5.6 Indexer Warmup and Attention Fitting

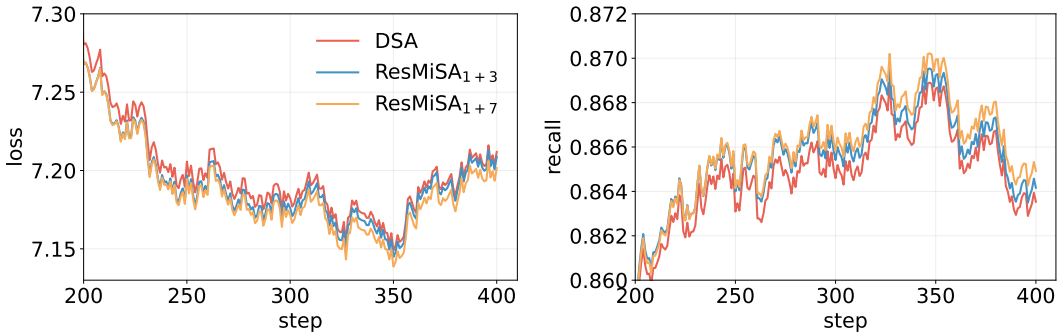
The shared expert also provides a direct gradient path through heads whose residuals are not selected. We examine this parameterization during indexer warmup, meaning indexer pretraining rather than a learning-rate schedule. All runs freeze the same dots-3-densemla (30B-A3B) pretrained backbone and train only its indexers to fit dense attention, using the same data, 512K sequences, 400 steps,

Table 2: Mean end-to-end TTFT in seconds (↓). The last two columns divide each baseline's TTFT by that of ResMiSA+LiteTopK.

Context	DSA	LiteTopK	ResMiSA+LiteTopK	vs. DSA	vs. LiteTopK
128K	46.53	47.13	40.93	1.14×	1.15×
256K	127.42	123.01	90.52	1.41×	1.36×
512K	397.05	364.52	213.66	1.86×	1.71×
1M	1358.91	1223.85	558.18	2.43×	2.19×

and a constant learning rate of 10^{-3}. The backbone continues to use its original dense and sliding-window attention, so this experiment measures attention fitting rather than downstream performance with sparse attention. Appendix G provides detailed settings.

We compare 32-head DSA with ResMiSA$_{1+3}$ and ResMiSA$_{1+7}$, which use ResMiSA's shared–residual rule with three or seven routed residual experts. All retain the same 32 parameterized indexer heads and trainable parameter count, but evaluate 32, 4, or 8 per-key scoring channels, respectively, with a token budget of 2048. Recall uses the per-head-normalized attention-mass metric defined in Appendix G.4, with $K = 2048$.



(a) Indexer cross-entropy.

(b) Recorded dense recall.

Figure 5: Indexer warmup with a frozen dots-3-densemla (30B-A3B) backbone. The plots show the last 200 of 400 training steps at 512K. Recall is sampled from training batches, not evaluated on a fixed validation set; its per-head-normalized definition is given in Appendix G.4.

Results. At the end of training, DSA, ResMiSA$_{1+3}$, and ResMiSA$_{1+7}$ have cross-entropies of 7.2119, 7.2088, and 7.2016, and recorded dense recalls of 86.35%, 86.42%, and 86.49%, respectively (Figure 5). Both variants achieve slightly lower fitting loss and higher training-batch recall at equal parameter counts and training steps, using $1/8$ and $1/4$ as many per-key scoring channels, respectively. These channel ratios do not measure training speedup. The results support effective attention fitting with sparse residuals. Single runs without a separate gradient-path ablation do not establish the cause of these small gains or improved downstream quality.

6 Conclusion

ResMiSA preserves all heads' linear contributions in one analytical shared expert and uses legal-key upper bounds to route sparse nonlinear corrections. Applied without additional training to DeepSeek-V3.2, this design reduces per-key scoring from 64 to eight channels, with RULER scores 0.06 percentage points below DSA on average and 0.21 points below at 128K. Combined with LiteTopK, it delivers 2.43× and 2.19× end-to-end TTFT speedups over DSA and LiteTopK alone at 1M context; task quality is evaluated separately over 4K–128K. Frozen-backbone indexer warmup also yields slightly lower fitting loss and higher training-batch recall with four or eight channels at the same parameter count and number of training steps. These results support shared–residual scoring for efficient indexing; downstream gains after indexer training remain to be established.

AI USE STATEMENT

In this work, we used generative AI tools to assist with drafting portions of the manuscript, polishing its language for clarity and readability, and generating code. All AI-assisted manuscript content and generated code were personally reviewed and checked by the authors before inclusion or use, and revised or corrected as needed. The authors take full responsibility for the final content of this work, including its text, claims, experimental results, code, and other artifacts, regardless of whether generative AI tools assisted in their creation.

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, 2024. URL https://arxiv.org/abs/2308.14508.

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. *arXiv preprint arXiv:2603.12201*, 2026. URL https://arxiv.org/abs/2603.12201.

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

Long Cheng, Ritchie Zhao, Timmy Liu, Mindy Li, Xianjie Qiao, Kefeng Duan, Yu-Jung Chen, Xiaoming Chen, Bita Darvish Rouhani, and June Yang. Guess-Verify-Refine: Data-aware top-k for sparse-attention decoding on blackwell via temporal correlation. *arXiv preprint arXiv:2604.22312*, 2026. URL https://arxiv.org/abs/2604.22312.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019. URL https://arxiv.org/abs/1904.10509.

Yaosheng Fu, Guangxuan Xiao, Xin Dong, Song Han, and Oreste Villa. SparDA: Sparse decoupled attention for efficient long-context llm inference. *arXiv preprint arXiv:2606.04511*, 2026. URL https://arxiv.org/abs/2606.04511.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024. URL https://arxiv.org/abs/2404.06654.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world github issues?, 2024. URL https://arxiv.org/abs/2310.06770.

Peng Jin, Bo Zhu, Li Yuan, and Shuicheng Yan. MoH: Multi-head attention as mixture-of-head attention. *arXiv preprint arXiv:2410.11842*, 2024. URL https://arxiv.org/abs/2410.11842.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2404.14469.

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. Deepseek-v3.2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025. URL https://arxiv.org/abs/2512.02556.

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. *arXiv preprint arXiv:2607.24593*, 2026. URL https://arxiv.org/abs/2607.24593.

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, et al. Moba: Mixture of block attention for long-context llms. *Advances in Neural Information Processing Systems*, 2025. URL https://www.arxiv.org/abs/2502.13189.

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

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

Yufei Xu, Fanxu Meng, Fan Jiang, Yuxuan Wang, Ruijie Zhou, Zhaohui Wang, Jiexi Wu, Zhixin Pan, Xiaojuan Tang, Wenjie Pei, et al. HISA: Efficient hierarchical indexing for fine-grained sparse attention. *arXiv preprint arXiv:2603.28458*, 2026. URL https://arxiv.org/abs/2603.28458.

Howard Yen, Tianyu Gao, Minmin Hou, Ke Ding, Daniel Fleischer, Peter Izsak, Moshe Wasserblat, and Danqi Chen. HELMET: How to evaluate long-context models effectively and thoroughly. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.02694.

Ziqi Yin, Jianyang Gao, Peiqi Yin, Jiangneng Li, and Gao Cong. LiteTopK: Exploiting the curse of dimensionality for a fused indexer-topk kernel in long-context sparse attention. *arXiv preprint arXiv:2607.11976*, 2026. URL https://arxiv.org/abs/2607.11976.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, et al. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025. URL https://arxiv.org/abs/2502.11089.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 2020. URL https://arxiv.org/abs/2007.14062.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chengxing Xie, Chenzheng Zhu, et al. GLM-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026. URL https://arxiv.org/abs/2602.15763.

Xiaofeng Zhang, Yikang Shen, Zeyu Huang, Jie Zhou, Wenge Rong, and Zhang Xiong. Mixture of attention heads: Selecting attention heads per token. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 2022. URL https://arxiv.org/abs/2210.05144.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H_2O: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in neural information processing systems*, 2023. URL https://arxiv.org/abs/2306.14048.

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. *arXiv preprint arXiv:2605.07363*, 2026. URL https://arxiv.org/abs/2605.07363.

A SCORE RECONSTRUCTION

We examine how well the shared expert reconstructs full-head DSA scores and how this fidelity improves as residual experts are added. The five configurations are $1+0$, $1+1$, $1+3$, $1+7$, and $1+15$, where the first term denotes the analytical shared expert and the second the number of routed residual experts. We use ten examples from each of `swe-bench` (Jimenez et al., 2024), `lsht`, `triviaqa_e`, and `narrativeqa` (Bai et al., 2024) (40 examples in total). For corresponding scalar scores a from DSA and b from a shared–residual configuration, we define

$$\mathrm{sim}(a,b) = \begin{cases} 1, & a = b = 0, \\ 1 - \dfrac{|a-b|}{|a|+|b|}, & \text{otherwise.} \end{cases} \tag{15}$$

This entrywise similarity lies in $[0,1]$ and captures agreement in both sign and relative magnitude. We compute similarity for each valid score entry before aggregating across sequences and layers. Within each dataset, all valid entries across all token positions and layers receive equal weight; masked entries are excluded. Let Ω_d contain these entries for dataset d, and let a_u and $b_u^{(r)}$ denote the corresponding scores. We report

$$S_d^{(r)} = \frac{1}{|\Omega_d|} \sum_{u \in \Omega_d} \mathrm{sim}(a_u, b_u^{(r)}), \qquad \bar{S}^{(r)} = \frac{1}{4} \sum_{d=1}^{4} S_d^{(r)}. \tag{16}$$

Here $d = 1, \ldots, 4$ indexes the four datasets listed above; their means receive equal weight in the overall average. This metric compares raw scalar indexer scores, not attention probabilities or attention mass recall.

Table 3: DSA score similarity (%) for one shared expert and varying residual budgets. Each dataset contributes ten examples. Avg. is the unweighted mean of the four dataset-level results.

Dataset	$1+0$	$1+1$	$1+3$	$1+7$	$1+15$
SWE	66.42	69.83	73.93	80.74	87.63
LSHT	65.48	70.84	75.55	81.07	87.82
TriviaQA	62.47	68.04	73.19	79.70	87.16
NarrativeQA	65.02	69.93	74.34	80.57	87.48
Avg.	64.85	69.66	74.25	80.52	87.52

The shared expert alone achieves 64.85% mean similarity to DSA scores (Table 3). Adding residual experts improves fidelity on every dataset, reaching 80.52% at $1+7$ and 87.52% at $1+15$. The latter doubles the per-key scoring channels for a 7.00-percentage-point gain; gains per additional channel decrease across the tested budgets. We therefore use $1+7$ as the default balance between fidelity and computation, also matching MISA's eight-channel configuration. Because entrywise score similarity does not guarantee preservation of token rankings, the main text evaluates attention mass recall and RULER performance separately.

B FUSED SCORING AND CANDIDATE FILTERING WITH LITETOPK

This appendix details how the shared and residual paths in Section 4.3 are integrated with LiteTopK. A single kernel scans keys, computes joint scores, filters candidates immediately, and refreshes filtering thresholds at window boundaries. This fused scan avoids materializing the full query–key score matrix. Final top-K selection then operates on the retained candidates in a separate stage.

B.1 INPUTS AND JOINT SCORING

For query i, the inputs are the shared vector q_i^{share}, the routed residual-head set A_i with its query vectors and signed weights, keys $\tilde{k}_j$, and the legal key set I_i. The shared vector and residual set are prepared as described in Sections 4.1 and 4.2, respectively. Joint scoring follows Equation 7:

$$\hat{s}_{i,j} = (q_i^{\text{share}})^\top \tilde{k}_j + \sum_{h \in A_i} w_{i,h} \max(-q_{i,h}^\top \tilde{k}_j, 0), \qquad j \in I_i. \tag{17}$$

Scaling factors are included in the key representation and effective weights following the convention in Section 3. The shared term is not passed through ReLU, residual weights retain their signs, and the final score is not clipped to zero.

The scan also takes prepared seed candidates, a seed histogram, an initial threshold, a selection budget K, a candidate capacity C, and a refresh-window length W. Seed candidates come from a previously processed key region, while the subsequent scan covers the remaining region. To unify bucketing and threshold comparisons, scores are mapped to a monotonically decreasing coordinate:

$$b_{i,j} = -\alpha_i(\hat{s}_{i,j} + o_i), \qquad \alpha_i > 0. \tag{18}$$

Here, o_i and α_i are per-query offset and scale parameters. Smaller coordinates correspond to higher scores; for finite coordinates and positive thresholds, $b_{i,j} < g_i$ denotes passing the current threshold g_i. These equations describe the mathematical operations, omitting specific floating-point rounding and storage representations.

B.2 FUSED SCAN ALGORITHM

Algorithm 1 scans contiguous key splits in parallel. Each split maintains its own histogram and threshold, while all splits append records to shared per-query candidate sets. Each histogram counts the seeds and candidates flushed from local buffers; it does not count every scored key or direct buffer-overflow append.

Algorithm 1 Fused scoring, candidate filtering, and subsequent selection

Require: Queries, keys, weights, and legal key sets in Equation 17
Require: Seed candidates, seed histogram, and initial threshold; o_i, α_i; budget K, capacity C, window length W
Ensure: Token index set $\mathcal{T}_i$ selected from retained candidates
Preparation (outside the kernel)
Initialize each query's candidate set $\mathcal{C}_i$ with seed candidates
Fused kernel
for all query groups and contiguous key splits, in parallel **do**
  Initialize local H_i and g_i from seed state; clear local candidate buffers
  for each key tile in the current split **do**
    Load keys; jointly compute shared- and residual-expert dot products
    for all query–key pairs in this tile with $j \in I_i$, in parallel **do**
      Compute the score by Equation 17 and $b_{i,j}$ by Equation 18
      if $b_{i,j} < g_i$ **then**
        Buffer the candidate; if the local buffer is full, append directly to $\mathcal{C}_i$ within capacity, without updating H_i
      end if
    end for
    if the current W-tile window is complete, or the split ends **then**
      Bucket buffered candidates by coordinate, add counts to H_i, and append candidates to $\mathcal{C}_i$ within capacity C
      Synchronize to complete the current window's statistics
      For each query, find the first bucket u with $\sum_{v=0}^{u} H_i[v] \geq K$
      if such a bucket exists and has a finite conservative upper edge e_u **then**
        Tighten the threshold: $g_i \leftarrow \min(g_i, e_u)$
      end if
      Synchronize and clear buffers; use the updated threshold in the next window
    end if
  end for
end for ▷ End of fused kernel
Final selection and output (outside the kernel)
Coarsely bucket each $\mathcal{C}_i$ and locate the boundary bucket whose cumulative count reaches K; retain all valid candidates if fewer than K remain
If a boundary bucket exists, retain candidates strictly better than it; refine the ordering boundary within it and resolve ties to fill the remaining slots, continuing selection-stage boundary-overflow processing if needed
If upstream index permutation was used, map selected indices back to original token indices to obtain $\mathcal{T}_i$

C Positive- and Negative-ReLU Zero Rates

Experimental setup. We compare the activation sparsity of the original positive-ReLU response and the negative-ReLU residual before head weighting or summation. We use ten examples from each of `lsht`, `triviaqa_e`, and `narrativeqa` (Bai et al., 2024). We examine all 61 layers of DeepSeek-V3.2 (Liu et al., 2025), with 64 indexer heads per layer, and all 78 layers of GLM-5 (Zeng et al., 2026), with 32 indexer heads per layer.

Zero-rate definition. Let Ω contain the measured sample–layer–query–key–head entries at legal positions, excluding masked entries. We pool counts across layers and define the positive- and negative-ReLU zero rates as

$$z_{\pm} = \frac{1}{|\Omega|} \sum_{u \in \Omega} \mathbf{1}[\mathrm{ReLU}(\pm x_u) = 0] \,. \tag{19}$$

An exactly zero dot product contributes to both counts: for the same finite-valued inputs and denominator, $z_+ + z_- = 1 + \Pr(x = 0)$. Thus, the two percentages can sum to slightly more than 100%; they are not renormalized.

D Ablation Studies

We examine three decisions in the residual branch: how to rank experts, how many to activate, and whether their identities should depend on the query. All studies measure attention mass recall by context position using the setup in Section 5.1; the routing comparison also reports RULER scores. Appendix A complements the residual-budget study with raw score similarity on four datasets.

D.1 Upper-Bound versus Mean-Pooling Routing

We retain one shared expert and seven residual experts and change only the residual-routing criterion. Figure 6 shows a modest advantage for upper-bound routing over mean pooling beyond approximately 20K positions. The RULER comparison in Table 4 also favors upper-bound routing, with a gain of 0.17 percentage points in average score and 0.44 points at 128K. The consistent advantage supports using extrema to protect against large local residuals that block averaging can obscure.

Table 4: RULER routing ablation (↑) for ResMiSA with one shared expert and seven residual experts. Only the routing criterion is varied.

Routing	4K	8K	16K	32K	64K	128K	Avg.
Mean pooling	96.43	96.21	96.02	95.91	94.79	93.00	95.39
Upper bound	**96.53**	**96.39**	**96.06**	**95.98**	**94.96**	**93.44**	**95.56**



Figure 6: Routing-criterion ablation for ResMiSA at the $1 + 7$ expert budget. Upper-bound routing improves attention mass recall over mean pooling at longer positions.

D.2 Residual Budget

We vary the number of residual experts from zero to 1, 3, 7, and 15 while retaining the same all-head shared expert, using full-head DSA as the reference. Here, the leading one always denotes the linear expert constructed from all 64 heads, and the second number counts the selected negative-ReLU residuals. Figure 7 shows an overall improvement in recall as the residual budget grows, with diminishing returns. The trend supports $1 + 7$ as a compromise between selection quality and the logical scoring budget.

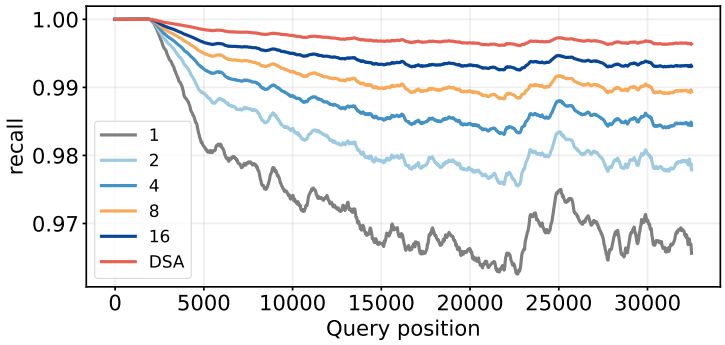


Figure 7: Residual-budget ablation for ResMiSA. Legend values count the total per-key scoring channels: 1 denotes the shared expert alone, while 2, 4, 8, and 16 denote one shared expert plus 1, 3, 7, and 15 residual experts, respectively; DSA uses all 64 indexer heads. Increasing the residual budget generally improves attention mass recall with diminishing returns. The shared expert incorporates the linear contributions of all 64 heads.

D.3 Dynamic versus Fixed Residual Heads

To test whether routing must adapt to each query, we compare dynamic $1 + 7$ selection with a fixed set of seven residual heads per layer. To avoid evaluating fixed heads on their calibration data, we calibrate on ten examples from the `lsht` subset of LongBench, separate from the SWE-bench evaluation examples. For each layer, we count how often each head is selected by dynamic top-seven routing, retain the seven most frequently selected heads, and freeze their indices during evaluation. Empirically, the layer-wise head selections stabilize with ten calibration examples. The queries, weights, and all-head shared vector remain input-dependent in both variants. Dynamic selection yields higher recall throughout the displayed position range (Figure 8), indicating that the residual budget benefits from adapting to the query even when the shared path remains input-dependent in both variants.

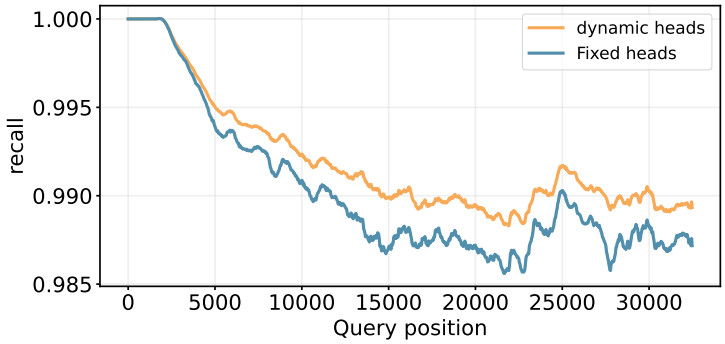


Figure 8: Dynamic versus fixed residual heads for ResMiSA. Query-dependent routing outperforms the layer-wise fixed $1 + 7$ layout in attention mass recall over the displayed position range. Fixed residual-head indices are calibrated on separate LongBench examples.

E END-TO-END TTFT ACROSS CONCURRENCY SETTINGS

Experimental setup. We follow the main-text TTFT protocol in Section 5.5, using ten warm-up runs followed by ten measured requests, with each request generating one output token. To assess whether the latency gains depend on the scheduler limit, we vary maximum concurrency over 1, 2, 4, 8, and 16 and report mean TTFT, holding all other settings fixed. At a maximum concurrency of 16 we issue sixteen measured requests instead of ten, so that the concurrency limit is actually reached.

Table 5: End-to-end TTFT (seconds) across context lengths and maximum concurrency settings. The last two columns report the speedup of ResMiSA+LiteTopK over DSA and LiteTopK, respectively. Maximum concurrency specifies the scheduler limit rather than a constant number of simultaneously active requests.

Context	Max Concurrency	DSA	LiteTopK	ResMiSA +LiteTopK	vs. DSA	vs. LiteTopK
128K	1	14.17	14.33	12.51	1.13×	1.15×
	2	25.99	26.31	22.87	1.14×	1.15×
	4	46.53	47.13	40.93	1.14×	1.15×
	8	71.25	74.31	62.74	1.14×	1.18×
	16	116.55	118.00	102.53	1.14×	1.15×
256K	1	38.73	37.15	27.56	1.41×	1.35×
	2	71.21	68.66	50.53	1.41×	1.36×
	4	127.42	123.01	90.52	1.41×	1.36×
	8	195.13	188.42	138.86	1.41×	1.36×
	16	320.11	309.55	226.83	1.41×	1.36×
512K	1	118.77	109.23	65.46	1.81×	1.67×
	2	221.78	203.69	121.09	1.83×	1.68×
	4	397.05	364.52	213.66	1.86×	1.71×
	8	607.46	557.90	330.74	1.84×	1.69×
	16	992.99	912.48	540.00	1.84×	1.69×
1M	1	406.53	364.88	168.09	2.42×	2.17×
	2	761.76	684.77	311.48	2.45×	2.20×
	4	1358.91	1223.85	558.18	2.43×	2.19×
	8	2062.91	1874.99	854.00	2.42×	2.20×
	16	3421.32	3063.45	1397.65	2.45×	2.19×

Results. Table 5 shows consistent acceleration across the tested concurrency settings. At 512K context, the absolute TTFT of ResMiSA+LiteTopK increases from 65.46 s to 540.00 s as maximum concurrency increases from 1 to 16, while its speedups remain stable at 1.81–$1.86\times$ over DSA and 1.67–$1.71\times$ over LiteTopK. At 1M, the corresponding speedups remain 2.42–$2.45\times$ and 2.17–$2.20\times$. The gains therefore persist across the tested scheduler limits, extending beyond the main-text setting of four.

F INDEXER COMPONENT TIMING

Experimental setup. We use NVIDIA Nsight Systems (`nsys`) to profile the three configurations in Section 5.5 under the same TTFT experimental settings, with a context length of 1M (1,048,576 tokens), one output token, and `max_concurrency=4`. All configurations use DeepSeek-V3.2 with 61 layers, vLLM tensor parallelism of eight, expert parallelism enabled, and 8192-token prefill chunks. For each configuration, we perform ten warm-up runs followed by ten measured requests and report cumulative GPU kernel time per request, averaged across the eight GPUs.

Table 6: Indexer component timing at 1M context, reported as cumulative GPU kernel time in seconds per request per GPU. LiteTopK uses the full DSA indexer; ResMiSA+LiteTopK uses one shared expert and seven residual experts. Fused path denotes the fused token-scoring and Top-K selection path, including associated kernels. Residual sel. and Shared constr. denote residual-expert selection and shared-expert construction. Other includes remaining indexer operations, which may involve KV movement, quantization, and cache writes. Totals include Other, and values are rounded independently.

DSA	**LiteTopK**	**ResMiSA+LiteTopK**
Token scoring: 231.39 s	Fused path: 262.93 s	Head scoring: 5.99 s
Token Top-K: 57.89 s		Residual sel.: 0.04 s
		Shared constr.: 1.07 s
		Fused path: 78.65 s
Other: 0.42 s	Other: 0.49 s	Other: 0.64 s
Total: 289.70 s	**Total: 263.42 s**	**Total: 86.40 s**

Component accounting. Head scoring denotes routing-score preparation, including key pooling, legal-interval boundary handling, and score repair. Shared-expert construction includes query weighting, reduction across heads, and FP8 quantization. Scoring and selection paths aggregate the associated kernels. Overall, indexer kernel time decreases from 289.70 s for DSA and 263.42 s for LiteTopK to 86.40 s for ResMiSA+LiteTopK, corresponding to $3.35\times$ and $3.05\times$ speedups, or reductions of 70.18% and 67.20%, respectively, calculated from the unrounded measurements. These measurements quantify indexer work as cumulative GPU kernel time; end-to-end TTFT additionally reflects the rest of the model and request scheduling, so the two quantities are reported separately.

G Detailed Settings for Indexer Warmup

This appendix specifies the frozen-backbone attention-fitting experiment in Section 5.6. It evaluates shared–residual scoring during indexer training on dots-3-densemla, separately from the training-free DeepSeek-V3.2 inference experiments.

G.1 Backbone and Indexers

All three runs use the same dots-3-densemla (30B-A3B) pretrained backbone and freeze both the backbone and its multi-token prediction (MTP) modules. Only the nine indexers attached to global-attention layers are trained; each such layer has 64 attention heads. Table 7 summarizes the shared indexer settings.

Table 7: Shared indexer settings for warmup.

Item	Setting
Indexers / heads per indexer	9 / 32
Head dimension	128
Total trainable parameters	22,563,072
Final token top-K	2048

G.2 Data and Optimization

All runs use the same 129 weighted data sources covering long-context material, Chinese and English books, web pages, papers, and code, with the dots tokenizer and no length upsampling. Over 400 steps, each run processes 3200 sequences, or approximately 1.678B fixed token slots. This total counts allocated sequence positions rather than deduplicated or non-padding tokens. Table 8 gives the shared optimization settings.

Table 8: Shared training and optimization settings for indexer warmup.

Item	Setting
Compute / parallelism	64 GPUs (8 nodes); TP1 / PP1 / CP32 / EP8 / DP2
Sequence length / training steps	524,288 tokens (512K) / 400
Micro / global batch / accumulation	1 / 8 / 4 microbatches per data-parallel replica per step
Learning rate / LR warmup steps	Constant 10^{-3} / 0
Optimizer	Distributed Muon (matrix parameters)
Muon settings	Momentum 0.95; Nesterov; 5 NS iterations
Adam branch	$\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-15}$
Weight decay / gradient clipping	0.1 / 1.0
Precision	BF16; FP32 master weights / main gradients
Attention / hidden dropout	0 / 0
Training / dataset seed	512 / 512

TP, PP, CP, EP, and DP denote tensor, pipeline, context, expert, and data parallelism, respectively; NS denotes Newton–Schulz.

G.3 Scoring Variants and Supervision

DSA uses weighted positive-ReLU scores from all 32 heads, whereas ResMiSA$_{1+3}$ and ResMiSA$_{1+7}$ combine an all-head shared linear term with three and seven negative-ReLU residuals, respectively; all three retain 32 parameterized indexer heads, with 32, 4, and 8 per-key scoring channels. ResMiSA performs query-wise upper-bound routing with 2048-token statistics blocks; actual scores retain the signs of the head weights and the scaling $1/\sqrt{32 \times 128}$. The comparison holds trainable capacity and the step budget fixed while varying per-key scoring computation; it does not measure training speedup.

Let $p_{i,j}$ be the frozen backbone's head-averaged attention probability over the full legal context (Equation 12) and $\pi_{i,j}$ the indexer's predicted probability. The training objective is

$$\mathcal{L}_{\mathrm{CE},i} = -\sum_{j \in I_i} p_{i,j} \log \pi_{i,j}. \tag{20}$$

The runs enable FlashKL with loss coefficient 1.0; the reported objective is cross-entropy, not KL divergence.

G.4 Recall and Scope of the Results

For indexer warmup, we report dense recall by computing a retained-mass ratio within each attention head and then averaging these ratios. This differs from the inference metric in Section 5.3, which averages head probabilities before constructing and normalizing by the reference selection. For query i, let $p_{i,a,j}$ be head a's softmax probability over all legal keys I_i, computed with the actual attention-logit scale, and let $\mathcal{S}_i$ be the indexer's selected set, shared across attention heads. With $K_i = \min(2048, |I_i|)$, define $\mathcal{O}_{i,a} = \mathrm{TopK}_{j \in I_i}(p_{i,a,j}, K_i)$. The query-level metric is

$$R_i^{\mathrm{dense}} = \frac{1}{H_{\mathrm{attn}}} \sum_{a=1}^{H_{\mathrm{attn}}} \frac{\sum_{j \in \mathcal{S}_i} p_{i,a,j}}{\sum_{j \in \mathcal{O}_{i,a}} p_{i,a,j}}. \tag{21}$$

Here $H_{\mathrm{attn}} = 64$ denotes the backbone's global attention heads, not the 32 indexer heads; all three runs use this definition. Recall is sampled from training-batch queries, not measured over a fixed validation set, and Figure 5 displays only the final 200 of 400 steps. The measurements support an attention-fitting comparison under the stated budget. Single runs with training-batch recall do not establish statistical significance, held-out generalization, or downstream task improvements.