DYNAMIC MASK ATTENTION: TRAINABLE SPARSE ATTENTION WITH FUSED BLOCK SCREENING

Anonymous authors
Paper under double-blind review

ABSTRACT

Dynamic sparse attention must keep selection overhead low to translate sparsity into GPU speedups. We introduce Dynamic Mask Attention (DMA), an algorithm–systems co-design that fuses learned block screening into IO-aware attention kernels. DMA uses a rank-1 query-key gate to derive cheap blockwise bounds on gate scores, reducing memory traffic and bypassing QK score computation and post-score operations on skipped tiles. This screening is integrated into forward, backward, and decoding kernels without materializing a dense routing matrix. The same gate modulates retained attention logits and is trained through the language-modeling objective. Across models ranging from 80M to 8B parameters, DMA improves the quality-efficiency trade-off in pre-training, long-context evaluation, and sparse adaptation, while maintaining quality comparable to full attention. On Qwen3-8B, DMA achieves an average score of 64.26 versus 62.50 for Full Attention across RULER-128K, LongBench, and benchmarks of reasoning, coding, and knowledge. It also delivers approximately 1.2$\times$ the end-to-end prefill and decode serving throughput of Full Attention under an 8K-input/24K-output workload.

1 INTRODUCTION

Long-context language models are increasingly used in workloads that require retrieving, integrating, and reasoning over information across thousands of tokens, such as long-document understanding, multi-turn reasoning, and repository-level code generation (Snell et al., 2024; Park et al., 2023; DeepMind, 2025; HuggingFace, 2025; Guo et al., 2025; Zhang et al., 2024; Team, 2025). However, standard multi-head self-attention (Vaswani et al., 2017) remains a fundamental bottleneck: it evaluates all query-key interactions, resulting in quadratic training cost (Zaheer et al., 2020) and growing KV-cache memory traffic during long-context inference (Kwon et al., 2023). Although FlashAttention-style kernels substantially improve IO efficiency and hardware utilization (Dao et al., 2022), they preserve the dense attention pattern.

Sparse attention therefore offers a principled way to reduce the long-context cost, provided that low-utility interactions can be identified before expensive KV movement and attention computation. Existing designs expose a persistent tension between adaptability, trainability, and hardware efficiency. **Fixed sparse patterns**, such as sliding windows, global tokens, or block-structured masks, impose the same input-agnostic connectivity across inputs, positions, and heads. Their regularity maps cleanly onto tiled attention kernels, but the same regularity can miss distant yet task-relevant evidence outside the predefined pattern (Beltagy et al., 2020; Zaheer et al., 2020; Child et al., 2019; Fu et al., 2025). **Dynamic methods** instead select tokens, blocks, or cache entries at runtime using importance scores, top-k rules, routers, or auxiliary selector modules (Tang et al., 2024a; Li et al., 2024; Zhang et al., 2023; Xiao et al., 2024; Yuan et al., 2025a; Lu et al., 2025; Lai et al., 2025; Gao et al., 2024; Zhang et al., 2025). However, many dynamic schemes introduce selection mechanisms that are difficult to optimize jointly with the language-modeling objective. Moreover, if importance estimation itself requires dense or near-dense query-key scoring, much of the cost that sparsity aims to avoid is already incurred. Furthermore, even when the selected pattern is sparse in principle, irregular token- or cache-level access patterns can map poorly onto modern tiled kernels, eroding the realized speedup through routing, masking, and memory-access overheads.

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.

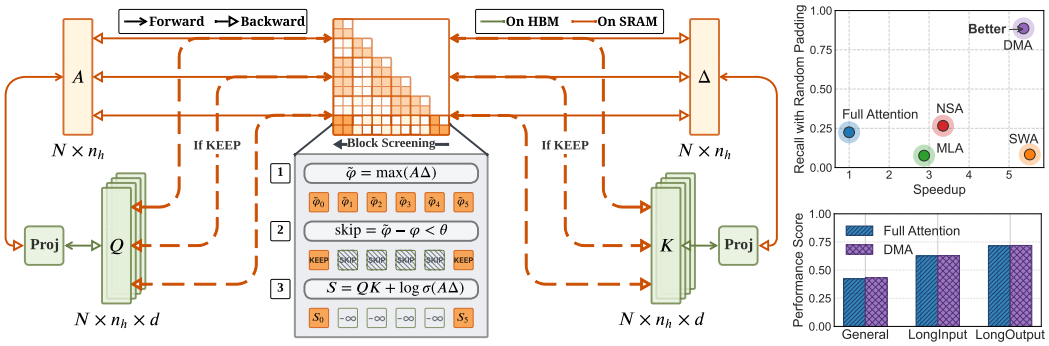

Figure 1: **Overview of Dynamic Mask Attention. Left**: DMA learns a head-specific gate from query and key features, uses a lightweight blockwise upper-bound test on this gate to screen candidate blocks before expensive memory movement and attention computation, and applies the retained gate signal as a smooth bias on attention scores. **Right**: DMA achieves a favorable speedup-recall trade-off and strong overall performance on long-context workloads among prior methods.

To simultaneously achieve adaptability, trainability, and hardware efficiency, an effective sparse attention for long contexts is naturally guided by three design goals: First, motivated by the observed division of labor in multi-head architectures, the selection signal should be **head-specific and input-adaptive**, rather than fixed by a hand-designed pattern or a single global sparsity rule (Voita et al., 2019; Clark et al., 2019; Schug et al., 2025). Second, the selection signal should be **low-overhead**: it should avoid materializing the full $\mathbf{QK}^\top$ matrix and add minimal overhead relative to the computation and memory traffic it prunes, enabling low-salience blocks to be skipped before KV-tile materialization (DeepSeek-AI; Jaber and Jaber, 2026; Gonçalves et al., 2026). Third, the induced sparsity should be **kernel-friendly**: it should align with the tile granularity of IO-aware attention kernels, so that pruning translates into regular blockwise scheduling and skipped KV-tile materialization rather than irregular masking overhead (Shah et al., 2024; Zadouri et al., 2026; Agarwal et al., 2026; Liu et al., 2026). Together, these goals raise a central question: ***Can we design a sparse-attention mechanism that combines head-specific, input-adaptive, and low-overhead selection with kernel-friendly execution?***

We introduce **Dynamic Mask Attention (DMA)**, a trainable sparse-attention mechanism that couples adaptive query-key relevance modeling with efficient tile-level execution. DMA starts from a *learned head-specific bilinear gate* over query-key features, which provides a flexible gate signal for modeling pairwise importance. To keep this signal lightweight, DMA adopts a rank-1 factorization of the bilinear gate. This factorization separates gate evaluation into query-side and key-side *projections* that can be computed independently and later combined to form pairwise gate scores, as illustrated in Fig. 1 left. As a result, DMA retains a learned head-specific gate signal while avoiding the cost of evaluating a full bilinear interaction for every query-key pair.

DMA applies this learned gate signal at the tile level, where sparse execution is performed. This tile-level design aligns with modern attention kernels and leverages *block-scale coherence* often observed in attention patterns (Romero et al., 2021; Arjovsky et al., 2016; Olsson et al., 2022; Jiang et al., 2024; Tang et al., 2024b; Xu et al., 2025). More importantly, the rank-1 gate admits a *cheap block-wise upper bound*: the largest possible gate score inside each candidate tile can be upper-bounded using precomputed query-side and key-side block statistics. This bound enables a *dynamic screening procedure* that scans candidate key-value tiles from near to far and *skips* tiles whose best possible gate score improves over a query-local running reference by less than the required margin. As a result, DMA avoids unnecessary KV-tile materialization and attention computation while keeping the sparse pattern directly executable by *IO-aware* tiled attention kernels.

The same gate also contributes a smooth non-positive bias to the logits of retained tiles, so the signal used for sparse execution is trained through the retained attention path rather than by a separate router or post-hoc selector. Sparsity therefore applies only to which tiles are executed; within every retained tile, DMA still performs standard token-level softmax attention over the full query-key dot products.

We evaluate DMA from both systems and modeling perspectives, integrating it into an IO-aware tiled attention kernel and testing its quality–efficiency trade-off across synthetic retrieval, pre-training, long-context benchmarks, and sparse adaptation of Qwen3-8B. Across model scales from 80M to 8B parameters, DMA achieves practical attention speedups while preserving dense-model quality

more reliably than competing sparse-attention baselines. The right panel of Fig. 1 previews this trade-off, showing DMA in the favorable high-speedup, high-recall regime while remaining close to full attention on aggregate performance. Our contributions are as follows:

- We introduce *Dynamic Mask Attention (DMA)*, a trainable sparse-attention mechanism that makes input-adaptive block-level execution decisions while preserving standard token-level softmax attention on retained tiles.
- We design a lightweight head-specific rank-1 bilinear gate together with a blockwise dynamic screening procedure, using the same learned signal to skip low-salience key-value tiles before materialization through a cheap upper-bound test and to modulate attention logits on retained tiles.
- We integrate DMA into an IO-aware tiled attention kernel and evaluate it across model scales from 80M to 8B parameters, demonstrating practical attention-module speedups and improved quality–efficiency trade-offs over strong sparse-attention baselines.

2 METHODOLOGY

Overview. Dynamic Mask Attention (DMA) augments IO-aware tiled attention with a learned sparse-execution path: it decides which key-value blocks to compute at the block level while preserving token-pair attention computation inside every retained block. Instead of deciding sparsity from the full attention matrix, DMA first introduces a lightweight token-level gate signal between queries and keys. This signal is then lifted to the block level by computing an upper bound on the gate values inside each query-key block pair. Based on this bound, DMA applies a relative screening rule to decide which key-value blocks should be retained before full tile materialization. Finally, on the retained support, DMA computes gated softmax attention with standard token-level logits and trains the gate through the retained attention path.

Notation. We describe one attention head and omit the batch dimension. Let $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N \times d_h}$ denote the query, key, and value matrices, with the usual scale $1/\sqrt{d_h}$ absorbed into $\mathbf{Q}$. The sequence is partitioned into blocks of size B, giving $T = \lceil N/B \rceil$ query and key-value blocks. We write I_i and J_j for the token indices in query block i and key block j.

2.1 RANK-1 GATED ATTENTION

DMA requires a gate that is expressive enough to capture head-specific query-key salience but cheap enough to evaluate before sparse savings are realized. A bilinear interaction is a natural choice for learning such a gate, since it directly scores query-key compatibility: $\widehat{Z}_{uv} = q_u^\top W k_v$, with $W \in \mathbb{R}^{d_h \times d_h}$. However, an unrestricted W would require dense bilinear evaluation over candidate token pairs, making the screening path nearly as expensive as the attention it is meant to prune. DMA therefore keeps the bilinear form but uses a rank-1 factorization, $W = w_A w_\Delta^\top$, $w_A, w_\Delta \in \mathbb{R}^{d_h}$, which gives

$$Z_{uv} = q_u^\top (w_A w_\Delta^\top) k_v = (q_u^\top w_A)(k_v^\top w_\Delta). \tag{1}$$

We define the scalar factors on the query side and on the key-side as $A_u = q_u^\top w_A$ and $\Delta_v = k_v^\top w_\Delta$, respectively. So we have $Z_{uv} = A_u \Delta_v$.

The gate Z_{uv} is not an approximation to the dot-product logit $q_u^\top k_v$; it is a learned salience feature used only for coarse block-level screening. Appendix E.1 visualizes the block-scale structure captured by this signal, and Appendix A gives a conditional justification: when block-averaged query and key statistics have a dominant mode, a rank-1 predictor yields a controlled approximation to block-level salience.

For retained token pairs, DMA injects this feature as a smooth suppressive bias:

$$G_{uv} = \log \sigma(Z_{uv}) = \log \sigma(A_u \Delta_v). \tag{2}$$

Since $\log \sigma(x) \leq 0$, the gate can only suppress unnormalized attention mass. Equivalently, adding G_{uv} before the softmax multiplies the unnormalized attention weight by $\sigma(Z_{uv}) \in (0, 1)$. This suppressive form is useful because it allows the model to down-weight low-salience interactions without introducing positive logit amplification that could destabilize attention.

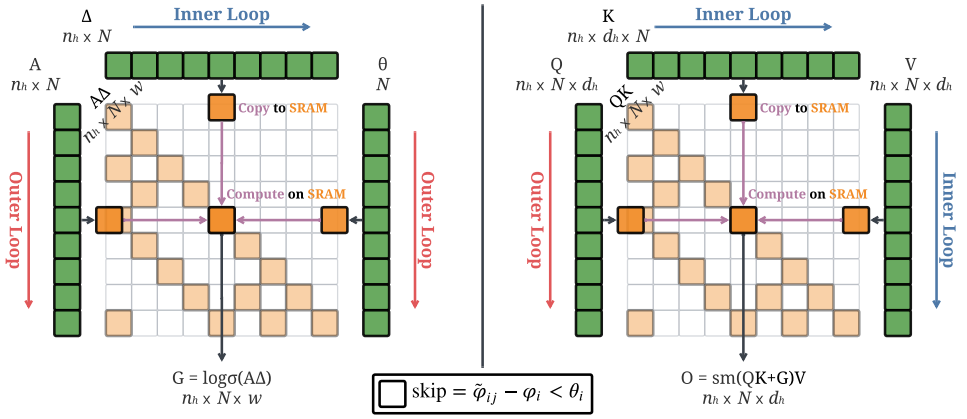

Figure 2: **Flow of Dynamic Mask Attention. Left**: DMA builds a block-level gate $\mathbf{G} = \log \sigma(\mathbf{A}\mathbf{\Delta}^\top)$ from projected scalar factors and skips blocks using the bound $\tilde{\varphi}_{ij} - \varphi_i < \theta_i$. **Right**: retained blocks are computed exactly as gated attention, $\mathbf{O} = \mathrm{softmax}(\mathbf{Q}\mathbf{K}^\top + \mathbf{G})\mathbf{V}$, in a FlashAttention-style tiled kernel.

2.2 Blockwise Gate Upper Bound

The rank-1 form in Eq. equation 1 enables inexpensive block-level screening. Before the tiled attention loop, DMA computes the scalar factors $\mathbf{A}$ and $\mathbf{\Delta}$. For a candidate query-key block pair (I_i, J_j), DMA summarizes the gate factors within the two blocks by their extrema:

$$\mathbf{A}_i^{\max} = \max_{u \in I_i} A_u, \qquad \mathbf{A}_i^{\min} = \min_{u \in I_i} A_u, \qquad \mathbf{\Delta}_j^{\max} = \max_{v \in J_j} \Delta_v, \qquad \mathbf{\Delta}_j^{\min} = \min_{v \in J_j} \Delta_v.$$

DMA then derives the blockwise gate upper bound

$$\tilde{\varphi}_{ij} = \max\{\mathbf{A}_i^{\max}\mathbf{\Delta}_j^{\max}, \mathbf{A}_i^{\max}\mathbf{\Delta}_j^{\min}, \mathbf{A}_i^{\min}\mathbf{\Delta}_j^{\max}, \mathbf{A}_i^{\min}\mathbf{\Delta}_j^{\min}\}, \tag{3}$$

since the product of two scalars over a rectangle attains its maximum at one of the four corners. Therefore, for every $(u, v) \in I_i \times J_j$, the gate signal is bounded above by $\tilde{\varphi}_{ij}$.

Since $\log \sigma(\cdot)$ increases monotonically, this also gives an upper bound on the gate bias:

$$G_{uv} = \log \sigma(A_u \Delta_v) \le \log \sigma(\tilde{\varphi}_{ij}). \tag{4}$$

This bound is lightweight, requiring only scalar block extrema and four scalar multiplications per candidate query-key block pair. It is specifically a gate-side bound: it does not upper-bound the full attention score $q_u^\top k_v + G_{uv}$, whose dot-product term has not yet been evaluated before tile materialization. DMA therefore uses $\tilde{\varphi}_{ij}$ as a learned block-salience signal for execution control, while retaining exact token-level dot-product attention on every retained tile.

2.3 Relative Block Screening

The blockwise bound $\tilde{\varphi}_{ij}$ estimates the strongest gate evidence that tile (i, j) can provide. Rather than comparing this score to a fixed global threshold, DMA uses a *query-local improvement criterion*. For each query block i, the kernel scans causal key blocks starting at the diagonal tile and proceeding toward position zero, so that the high-scoring local tiles typically encountered first establish a strong reference early. The kernel maintains a running reference φ_i, defined as the largest blockwise gate bound among tiles retained so far. We initialize $\varphi_i = -\infty$, so that the first visited causal tile is always retained.

Given a position-aware margin

$$\theta_i = \tau \cdot \frac{iB}{N}, \tag{5}$$

where τ controls the aggressiveness of sparsity, DMA retains a candidate tile only if its upper-bound gate score improves over the current reference by at least θ_i. Equivalently, the tile is skipped when

$$\textsc{skip}(i, j) \iff \tilde{\varphi}_{ij} - \varphi_i < \theta_i. \tag{6}$$

Only when the candidate tile is retained is the reference updated to $\varphi_i \leftarrow \max\{\varphi_i, \tilde{\varphi}_{ij}\}$.

This rule selects margin-separated improvements in gate evidence for each query block. A tile is not retained merely because its score is large in absolute terms; it must provide sufficient additional gate evidence relative to the strongest retained block so far. As a result, the screening threshold adapts to the local gate scale of each head and query block, whereas the resulting sparse pattern remains block-structured and directly executable inside an IO-aware tiled attention kernel.

2.4 Sparse execution with end-to-end gradients

Forward pass. Algorithm 1 gives the complete pseudocode. Let $\mathcal{R}(i)$ denote the retained key blocks for query block i. For each $j \in \mathcal{R}(i)$, DMA loads $\mathbf{K}_j$ and $\mathbf{V}_j$ and computes the full token-level dot-product scores, and adds the gate bias mentioned in Eq. equation 2

$$\mathbf{S}_{ij} = \mathbf{Q}_i \mathbf{K}_j^\top + \log \sigma(\mathbf{A}_i \mathbf{\Delta}_j^\top). \tag{7}$$

The diagonal tile receives the standard causal mask. The retained scores are then accumulated with the usual FlashAttention-style online softmax over the retained key blocks:

$$\mathbf{O}_i = \sum_{j \in \mathcal{R}(i)} \mathrm{softmax}_{j \in \mathcal{R}(i)}(\mathbf{S}_{ij}) \mathbf{V}_j. \tag{8}$$

Therefore, DMA changes only the set of tiles entering the attention computation. Within every retained tile, dot products, gate values, softmax normalization, and value aggregation are computed exactly for the truncated retained set.

Backward pass. DMA treats the binary skip decision as an execution decision and does not differentiate through the inequality in Eq. equation 6. The backward kernel uses key-major traversal and a key-side running reference φ_j to accumulate $\mathbf{dK}$, $\mathbf{dV}$, and $\mathbf{d\Delta}$ efficiently. Because forward screening instead uses a query-side reference φ_i, the two passes can retain different tiles. Backward screening therefore defines a traversal-dependent gradient approximation rather than an exact backward pass on the forward-retained support. Tiles skipped by the backward pass receive no direct gradient contribution from that pass.

For a tile retained by backward screening, let $\mathbf{D}^S_{ij}$ denote the score-gradient estimate reconstructed from the stored forward output and log-partition statistic, as in Algorithm 2. Since $\mathbf{G}_{ij} = \log \sigma(\mathbf{Z}_{ij})$, $\mathbf{Z}_{ij} = \mathbf{A}_i \mathbf{\Delta}_j^\top$, the corresponding gate-gradient estimate uses the derivative of $\log \sigma$:

$$\mathbf{D}^Z_{ij} = \mathbf{D}^S_{ij} \circ \sigma(-\mathbf{Z}_{ij}). \tag{9}$$

The scalar-factor gradient estimates are accumulated as

$$\mathbf{dA}_i \mathrel{+}= \mathbf{D}^Z_{ij} \mathbf{\Delta}_j, \qquad \mathbf{d\Delta}_j \mathrel{+}= (\mathbf{D}^Z_{ij})^\top \mathbf{A}_i. \tag{10}$$

These estimates then backpropagate through $\mathbf{A}_u = q_u^\top w_{\mathbf{A}}$ and $\mathbf{\Delta}_v = k_v^\top w_{\mathbf{\Delta}}$, updating both the gate directions $w_{\mathbf{A}}, w_{\mathbf{\Delta}}$ as well as the upstream query and key projections. The continuous gate is thus trained through the language-modeling objective using this approximate backward procedure. Algorithm 2 gives the complete pseudocode.

The backward kernel constructs its support online from the same scalar factors, blockwise bounds, and threshold parameters under its own traversal, avoiding storage of an explicit forward routing mask. Appendix E.4 measures forward–backward support overlap and gradient error relative to the screening-disabled gated operator.

Although backward-skipped tiles do not provide direct gradients, the shared gate factors allow retained interactions to provide an indirect corrective signal for future screening decisions. Because $\mathbf{A}_i$ and $\mathbf{\Delta}_j$ are shared across many retained interactions, this sharing induces a form of self-correction in the gate predictor. When the current gate is overly aggressive, retained blocks elsewhere in the sequence can still backpropagate prediction errors through $\mathbf{dS}_{ij}$ and $\mathbf{dG}_{ij}$ into the shared factors. These updates can increase future upper bounds for similar query-key patterns, making the corresponding blocks more likely to enter the retained set. Appendix C provides a conditional reconstruction-based analysis for forward-skipped blocks in terms of their dense probability mass; it does not guarantee that backward screening omits the same blocks.

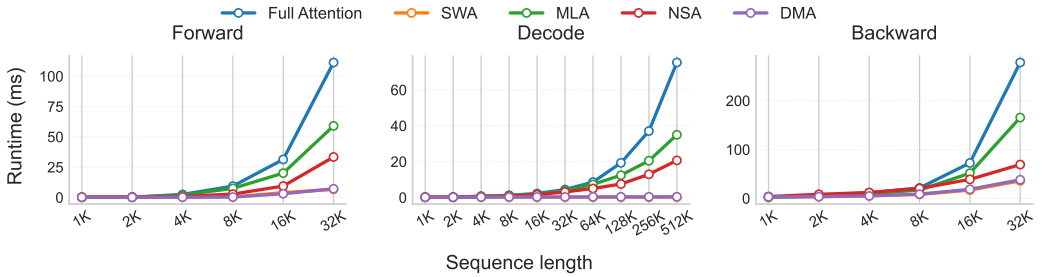

Figure 3: **Module performance**. This benchmark measures whether DMA's block screening yields practical runtime gains at the module level. On an NVIDIA A100 GPU, DMA delivers consistent speedups in forward pass, decoding, and backward pass, showing that dynamic sparsity can remain highly hardware-efficient.

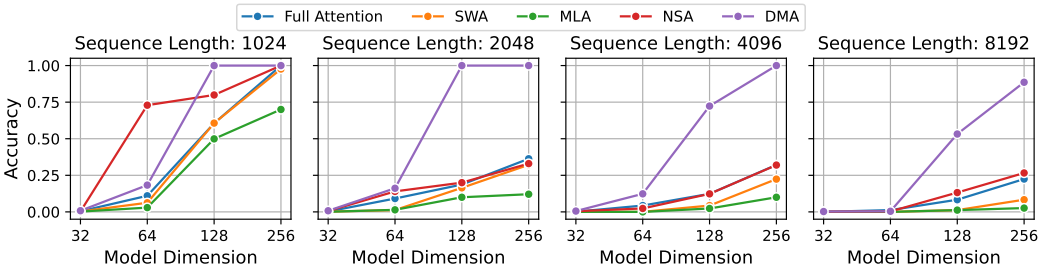

Figure 4: **Multi-Query Associative Recall**. This benchmark stresses long-range retrieval under strong interference using longer sequences, smaller model dimensions, and random noise padding between relevant spans. DMA remains strong in this setting, indicating that it can recover multiple target associations without relying on fixed local patterns.

3 Experiments

Evaluation overview. We structure the experiments around two central questions: whether DMA remains efficient at the attention-module level, and whether it improves long-context modeling quality rather than merely reduce computation. We therefore proceed in five stages. We first benchmark module end-to-end runtime to test whether DMA's block screening yields practical acceleration once projection and screening overheads are included. We then evaluate a harder multi-query associative recall task to isolate long-range retrieval under strong interference. Third, we study scaling laws to quantify the trade-off between perplexity and training compute across model sizes. Fourth, we evaluate the 1.7B models selected from the scaling-law study on standard benchmarks, longer-context benchmarks, and Needle-in-the-Haystack retrieval. Finally, we adapt the open-source Qwen3-8B dense backbone into a sparse-attention model through continued training and compare attention variants on long-input and long-output benchmarks.

Experimental settings. We evaluate all methods under matched model, data, and compute budgets, changing only the attention mechanism and its associated sparse-attention hyperparameters. Unless otherwise specified, DMA uses a 2K-step dense warm-up before sparse screening is activated to avoid early gate miscalibration. Shared implementation details, baseline choices, and tuning protocols are summarized in Appendix D.1; experiment-specific configurations are provided in Appendix D.

Module acceleration. This experiment tests the systems claim of DMA: the blockwise screening rule should translate into actual module savings rather than only theoretical sparsity. We benchmark end-to-end forward, decoding, and backward runtime of the attention module under the matched configurations in Appendix D.2. Figure 3 shows that DMA is highly competitive across all three stages and delivers the strongest overall runtime among the compared sparse methods. Its advantage comes from fusing the screening path directly into the attention kernel, so block skipping avoids substantial preprocessing overhead while reducing score, softmax, and memory-transfer work. Despite using dynamic sparsity, DMA remains close to the efficiency of SWA while substantially outperforming the other sparse baselines, indicating that its sparsity rule is compatible with practical end-to-end module acceleration.

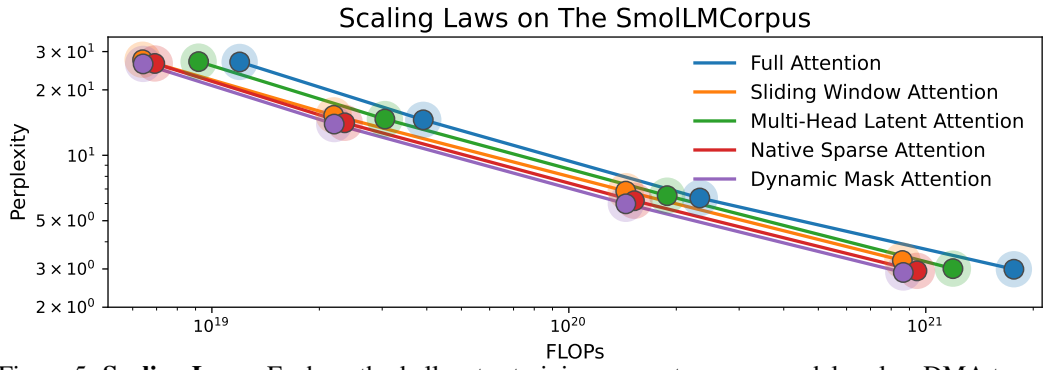

Figure 5: **Scaling Laws**. Each method allocates training compute across model scales. DMA traces the strongest perplexity-FLOPs frontier, matching full-attention quality with lower training compute and outperforming alternative sparse designs at comparable budgets.

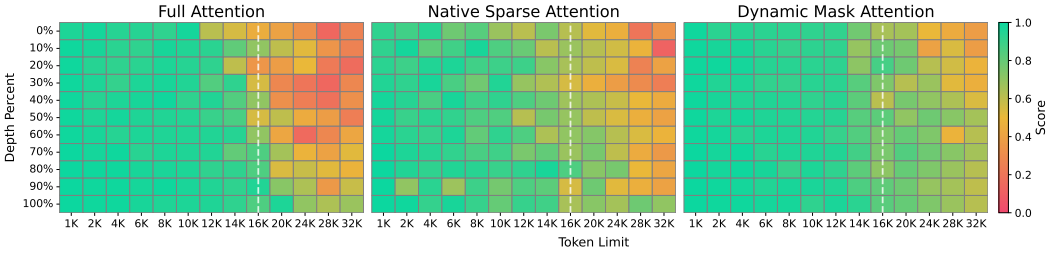

Figure 6: **Needle-in-the-Haystack performance**. Needle-in-the-Haystack measures whether a model can recover a short target span inserted at varying depths inside long documents. DMA preserves stronger retrieval accuracy across insertion depths and token limits, especially beyond the nominal context regime where length extrapolation becomes more challenging. The white vertical dashed line marks the model sequence length.

Associative recall. This benchmark isolates the retrieval regime in which DMA should matter most: relevant evidence is far away, interference is strong, and fixed local patterns are insufficient. We use the harder multi-query associative recall setup described in Appendix D.3, matching sparse methods by effective retained-key budget. Figure 4 shows that DMA remains strong across sequence lengths, indicating that it retains the relevant associations while remaining robust to substantial noise.

Scaling laws. We next test whether DMA's gains persist under compute-optimal pre-training by scaling models from 80M to 1.7B parameters on SmolLMCorpus; details are in Appendix D.4. Figure 5 reports perplexity versus training FLOPs. DMA traces a strong PPL–FLOPs frontier, largely matching full attention with lower compute and outperforming static or discrete sparse baselines at comparable budgets. This supports DMA's design: a jointly trained continuous gate coupled with block-level sparse execution improves the quality–compute trade-off across scales.

Pre-training benchmark evaluation. We next ask whether the advantage seen in scaling laws transfers to standard downstream evaluation. We evaluate the largest 1.7B models selected from the scaling-law study under the zero-shot and five-shot protocol in Appendix D.5. Table 1 shows that DMA attains the best average performance while remaining competitive across individual tasks in both evaluation modes. Under this matched pre-training setup, the results suggest that learning the sparse pattern during training can preserve downstream quality more effectively than introducing sparsification only at inference time.

Fine-tuned benchmark evaluation. We then evaluate the 1.7B models after 16K fine-tuning using the protocol in Appendix D.5. In Table 2, DMA has the highest mean score among the sparse methods when averaged across the seven tasks, close to Full Attention. DMA has the highest mean GSM8K score, while Full Attention has higher means on the other six tasks, including MBPP and LongBench. The reported standard deviations measure decoding sensitivity for fixed checkpoints, so small differences in task means do not establish statistical superiority. These results support comparable downstream quality after extending the context regime.

Extrapolated content retrieval. Needle-in-the-Haystack isolates a complementary question: can DMA preserve precise retrieval once the input length extends beyond the regime seen in pre-training?

Table 1: **Standard Benchmark Results**. This benchmark suite covers language modeling, commonsense reasoning, reading comprehension, and world knowledge for 1.7B models pre-trained from scratch at 8K context length. We report zero-shot and five-shot results. Under the same pre-training setup, DMA achieves the best average performance while remaining competitive with Full Attention and other sparse baselines across tasks.

Model	LAMBADA		MMLU		TriviaQA		ARC		PIQA		HellaSwag		OBQA		WinoGrande		Avg	
	acc ↑		acc ↑		acc ↑		acc ↑		acc ↑		acc ↑		acc ↑		acc ↑		acc ↑	
	0-shot	5-shot	0-shot	5-shot	0-shot	5-shot	0-shot	5-shot	0-shot	5-shot	0-shot	5-shot	0-shot	5-shot	0-shot	5-shot	0-shot	5-shot
Full Attention	44.3	40.4	35.4	36.8	**9.4**	13.2	53.4	**56.8**	72.9	73.2	56.1	56.8	**37.0**	38.0	57.3	58.6	45.7	46.7
H2O	44.2	38.9	34.8	35.7	7.4	10.2	53.3	56.6	72.8	73.2	55.6	56.4	36.6	37.8	56.9	58.1	45.2	45.9
InfLLM	44.2	**41.3**	35.1	35.9	8.0	11.7	53.1	56.7	72.4	73.3	55.8	56.1	36.6	38.0	56.8	57.7	45.2	46.3
Quest	43.9	40.9	35.1	36.1	7.6	10.9	53.1	56.2	72.6	73.2	56.1	55.8	36.8	37.9	57.2	58.2	45.3	46.1
DAM	44.5	41.2	34.6	35.2	8.9	**13.3**	52.1	55.1	72.3	71.0	56.2	54.4	36.3	38.0	56.0	57.2	45.1	45.7
Exact-Top	44.4	39.7	35.3	36.4	9.2	13.1	53.3	56.3	72.8	73.4	56.0	56.5	36.8	38.2	57.0	58.5	45.6	46.5
NSA	45.2	39.6	33.8	34.6	8.7	12.5	53.1	56.1	72.8	76.0	**56.7**	**58.9**	36.3	39.2	57.8	58.3	45.6	46.9
DMA (ours)	**45.9**	40.9	**37.0**	**38.2**	9.1	12.6	**55.6**	56.4	**73.4**	**76.6**	56.4	58.7	36.5	**39.6**	**58.4**	**60.4**	**46.5**	**47.9**

Table 2: **Downstream Benchmark Results**. We evaluate the 1.7B fine-tuned checkpoints using seeds 0, 1, and 2, with temperature 0.6 and top-p 0.95 for sampled generation. Each task reports the mean with its standard deviation as a subscript. Avg is the arithmetic mean of the seven task means.

Method	MMLU acc ↑	BBH acc ↑	GSM8K acc ↑	MATH acc ↑	MBPP acc ↑	LongBench acc ↑	RULER acc ↑	Avg acc ↑
Full Attention	$\mathbf{46.4}_{\pm 0.0}$	$\mathbf{38.2}_{\pm 0.004}$	$\underline{46.1}_{\pm 0.009}$	$\mathbf{12.2}_{\pm 1.17}$	$\mathbf{41.2}_{\pm 2.23}$	$\mathbf{30.3}_{\pm 0.037}$	$\mathbf{60.8}_{\pm 0.63}$	**39.31**
H2O	$42.4_{\pm 0.0}$	$34.1_{\pm 0.007}$	$44.8_{\pm 0.011}$	$9.8_{\pm 1.62}$	$36.1_{\pm 3.25}$	$28.1_{\pm 0.112}$	$47.3_{\pm 0.98}$	34.66
InfLLM	$44.2_{\pm 0.0}$	$36.2_{\pm 0.013}$	$45.2_{\pm 0.012}$	$10.1_{\pm 1.79}$	$37.5_{\pm 2.78}$	$28.2_{\pm 0.092}$	$54.7_{\pm 0.75}$	36.59
Quest	$44.0_{\pm 0.0}$	$34.1_{\pm 0.006}$	$45.2_{\pm 0.007}$	$11.3_{\pm 1.09}$	$38.2_{\pm 2.56}$	$29.1_{\pm 0.058}$	$51.1_{\pm 0.65}$	36.14
DAM	$45.0_{\pm 0.0}$	$35.2_{\pm 0.011}$	$45.1_{\pm 0.017}$	$10.2_{\pm 1.87}$	$34.6_{\pm 3.36}$	$25.1_{\pm 0.124}$	$48.7_{\pm 1.11}$	34.84
SeerAttention	$44.4_{\pm 0.0}$	$37.5_{\pm 0.012}$	$45.0_{\pm 0.014}$	$10.3_{\pm 1.57}$	$37.2_{\pm 1.72}$	$\underline{30.2}_{\pm 0.180}$	$58.7_{\pm 0.82}$	37.61
FlexPrefill	$43.6_{\pm 0.0}$	$35.4_{\pm 0.074}$	$44.7_{\pm 0.012}$	$7.4_{\pm 1.72}$	$22.2_{\pm 3.43}$	$27.7_{\pm 0.172}$	$60.2_{\pm 0.89}$	34.46
Exact-Top	$45.1_{\pm 0.0}$	$37.3_{\pm 0.004}$	$45.3_{\pm 0.009}$	$\underline{11.7}_{\pm 1.19}$	$38.9_{\pm 2.32}$	$28.6_{\pm 0.032}$	$45.2_{\pm 0.67}$	36.01
NSA	$43.8_{\pm 0.0}$	$38.0_{\pm 0.006}$	$45.7_{\pm 0.010}$	$11.2_{\pm 1.41}$	$40.2_{\pm 2.30}$	$29.7_{\pm 0.093}$	$59.2_{\pm 0.71}$	38.26
DMA (ours)	$\underline{46.2}_{\pm 0.0}$	$\underline{38.1}_{\pm 0.006}$	$\mathbf{46.5}_{\pm 0.011}$	$\underline{11.7}_{\pm 1.28}$	$\underline{41.0}_{\pm 2.13}$	$30.1_{\pm 0.024}$	$\underline{60.7}_{\pm 0.71}$	$\underline{39.19}$

Table 3: **Long-Context Benchmark Results**. Results on Qwen3-8B after sparse adaptation. RULER is evaluated at 128K, and LongBench uses the YaRN-extended 128K context configuration; both columns report macro averages over subtask means from three decoding seeds. Complete subtask results and evaluation settings are in Appendix D.8. TTFT and TPOT report prefill and decode serving throughput under the serving setup in Appendix D.7.

Method	TTFT tok/s ↑	TPOT tok/s ↑	RULER-128K acc ↑	LongBench acc ↑	MATH500 acc ↑	AIME 2024 acc ↑	LiveCodeBench acc ↑	GPQA acc ↑	Avg acc ↑
				Qwen3-8B					
Full Attention	47.7K	0.46K	58.18	**30.92**	95.9	**75.0**	**53.8**	61.2	62.50
HySparse	56.1K	**0.56K**	45.56	27.04	78.7	33.3	49.4	57.6	48.60
BLASST	54.6K	0.50K	65.92	29.44	92.4	66.7	51.6	61.2	61.21
DMA (ours)	**56.7K**	0.54K	**67.26**	30.60	**96.4**	73.3	53.3	**64.7**	**64.26**

Under the length-extrapolation setup in Appendix D.6, DMA's advantage widens as context length increases. In particular, once the input length exceeds the pre-training context range, all methods degrade, but DMA deteriorates much more slowly than the baselines. Figure 6 thus supports the view that DMA yields stronger length extrapolation.

Sparse adaptation via continued training. Finally, we adapt the dense Qwen3-8B backbone through continued training at 32K and evaluate long-context quality with YaRN extension to 128K. Appendix D.7 gives the adaptation recipe, evaluation protocol, and serving setup. Table 3 shows that DMA remains close to Full Attention on RULER-128K and LongBench. DMA also remains competitive on reasoning, coding, and knowledge benchmarks and improves prefill and decode throughput under the reported serving setup. Complete subtask results in Appendix D.8 show that DMA's mean scores exceed both sparse baselines on 19 of 21 LongBench subtasks and 10 of 13 RULER subtasks. These results support effective sparse adaptation and length-extrapolation quality on this backbone.

Ablation. The threshold sweep shows a controllable quality–sparsity trade-off (Appendix E.2). A layer- and GQA-position analysis further shows that retrieval retains more attention computation than QA and that sparsity varies across depth and query-head positions (Table 10 in the appendix). Additional ablations on Qwen3-8B yield three key findings, with kernel measurements taken from first-layer states. First, at approximately 62% sparsity and 8K context, the learned rank-1 selector achieves 99.13% full-QK Recall@K versus 49.46% for L2 scoring, with full-model RULER scores of 92.52 versus 74.61. Increasing the rank to 16 adds only 0.71 percentage points of recall and

0.30 mean RULER points, while increasing forward/backward latency by approximately 45%/41%, supporting rank 1 as an efficient operating point (Appendix E.3). Second, forward–backward mask IoU ranges from 97.91% to 99.82% across 1K–8K contexts and three adaptation stages, showing high support overlap while quantifying the traversal-dependent backward approximation (Appendix E.4). Third, replacing the selected 128×32 forward and 32×128 backward tiles with 32×32 tiles adds 0.62 percentage points of recall but raises forward/backward kernel latency to $1.90\times$/$1.93\times$ that of the selected geometry. Mean RULER scores span just 0.19 points across the four tested geometries, supporting the selected tiles' efficiency on A100 (Appendix E.5).

4 Related Work

DMA is a trainable sparse-attention mechanism that preserves the standard softmax-attention interface while integrating blockwise sparse execution into an IO-aware tiled kernel. We position it relative to four neighboring lines of work: sparse attention, hardware-aware attention kernels, alternative long-context operators, and KV-cache optimization systems.

Sparse attention. Early efficient-Transformer variants use fixed sparse patterns such as local windows, structured sparsity, or global tokens (Child et al., 2019; Beltagy et al., 2020; Zaheer et al., 2020), with sliding-window attention remaining attractive because of its simplicity and hardware friendliness (Fu et al., 2025). Later methods make sparsity data-dependent through retrieval scores, importance estimates, cache heuristics, or routing-oriented block selection (Tang et al., 2024a; Li et al., 2024; Zhang et al., 2023; Xiao et al., 2024; Desai et al., 2024; Zhang et al., 2025; Gao et al., 2024; Yuan et al., 2025a; Lu et al., 2025; Lai et al., 2025). DMA is closest to this dynamic sparse-attention family, but replaces hard routing with a smooth gate learned jointly with the model and converts it into block skipping through a lightweight upper-bound test inside the attention kernel.

Hardware-aware attention kernels. FlashAttention showed that tiled execution and online softmax can reduce attention memory movement while preserving exact dense attention (Dao et al., 2022), and subsequent systems improved memory management, programmability, and serving efficiency for large-scale attention workloads (Kwon et al., 2023; Dong et al., 2024; Ye et al., 2025). DMA is complementary to this line of work: it inherits the IO-aware tiled execution template, but augments the tile loop with learned block screening before expensive key-value loads.

Alternative long-context operators. Another route to long-context efficiency replaces softmax attention with alternative sequence operators, including linear attention and state-space models (Katharopoulos et al., 2020; Gu and Dao, 2023; Dao and Gu, 2024). DMA takes a more conservative path: it preserves the standard attention interface and exact token-token computation on retained tiles, targeting settings where precise retrieval and in-context reasoning remain important (Arora et al., 2024; Liu et al., 2024a).

KV-cache optimization systems. KV-cache methods reduce long-context inference cost by pruning, compressing, retrieving, or offloading cached states during decoding (Zhang et al., 2023; Li et al., 2024; Tang et al., 2024a; Xiao et al., 2024; Liu et al., 2024b; Huang et al., 2025). These systems primarily optimize which KV states are stored or moved, whereas DMA reduces the computation and memory traffic needed to evaluate attention within each layer; the two directions are therefore complementary and can in principle be combined.

5 Conclusion

We introduced Dynamic Mask Attention (DMA), a sparse-attention mechanism that learns a continuous query-key gating signal and converts it into blockwise skipping inside an IO-aware tiled attention kernel. DMA addresses the tension between adaptive long-context modeling and hardware-efficient execution: the learned gate modulates retained attention logits, while a cheap block-level upper-bound test skips low-salience tiles before materialization. Retained tiles still use standard token-level softmax attention, so DMA reduces execution cost without replacing the attention operator. Across module benchmarks, scaling-law pretraining, downstream evaluation, long-context retrieval, and sparse adaptation of Qwen3-8B, DMA improves the quality-efficiency trade-off relative to fixed and dynamic sparse baselines under matched settings. These results suggest that jointly trained, hardware-aware sparse execution is a practical direction for efficient long-context language modeling.

AI USE STATEMENT

We used generative AI tools to assist with drafting and language editing of the manuscript, including improving grammar, wording, paragraph flow, and the clarity of technical descriptions. The authors reviewed and revised the AI-assisted text and take full responsibility for the final manuscript, including its technical claims and experimental results.

REFERENCES

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22, 2023.

Google DeepMind. Gemini 2.5 pro, March 2025. URL `https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025`.

HuggingFace. Open r1: A fully open reproduction of deepseek-r1, 2025. URL `https://github.com/huggingface/open-r1`.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Kechi Zhang, Jia Li, Ge Li, Xianjie Shi, and Zhi Jin. Codeagent: Enhancing code generation with tool-integrated agent systems for real-world repo-level coding challenges. *arXiv preprint arXiv:2401.07339*, 2024.

Qwen Team. Qwen3, April 2025. URL `https://qwenlm.github.io/blog/qwen3`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

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*, 33:17283–17297, 2020.

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.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: fast and memory-efficient exact attention with io-awareness. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, 2022.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

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*, 2024a.

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*, 37:22947–22970, 2024.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710, 2023.

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. *arXiv preprint arXiv:2402.04617*, 2024.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, YX Wei, Lean Wang, Zhiping Xiao, et al. Native sparse attention: Hardware-aligned and natively trainable sparse attention. *arXiv preprint arXiv:2502.11089*, 2025a.

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. *arXiv preprint arXiv:2502.13189*, 2025.

Xunhao Lai, Jianqiao Lu, Yao Luo, Yiyuan Ma, and Xun Zhou. Flexprefill: A context-aware sparse attention mechanism for efficient long-sequence inference, 2025. URL https://arxiv.org/abs/2502.20766.

Yizhao Gao, Zhichen Zeng, Dayou Du, Shijie Cao, Peiyuan Zhou, Jiaxing Qi, Junjie Lai, Hayden Kwok-Hay So, Ting Cao, Fan Yang, et al. Seerattention: Learning intrinsic sparse attention in your llms. *arXiv preprint arXiv:2410.13276*, 2024.

Hanzhi Zhang, Heng Fan, Kewei Sha, Yan Huang, and Yunhe Feng. Dam: Dynamic attention mask for long-context large language model inference acceleration, 2025. URL https://arxiv.org/abs/2506.11104.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, 2019.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, 2019.

Simon Schug, Seijin Kobayashi, Yassir Akram, Joao Sacramento, and Razvan Pascanu. Attention as a hypernetwork. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=V4K9h1qNxE.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence.

Jaber Jaber and Osama Jaber. Streamindex: Memory-bounded compressed sparse attention via streaming top-k, 2026. URL https://arxiv.org/abs/2605.02568.

Nuno Gonçalves, Hugo Pitorro, Vlad Niculae, Edoardo Ponti, Lei Li, Andre Martins, and Marcos Treviso. Adasplash-2: Faster differentiable sparse attention, 2026. URL https://arxiv.org/abs/2604.15180.

Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. Flashattention-3: Fast and accurate attention with asynchrony and low-precision, 2024. URL https://arxiv.org/abs/2407.08608.

Ted Zadouri, Markus Hoehnerbach, Jay Shah, Timmy Liu, Vijay Thakkar, and Tri Dao. Flashattention-4: Algorithm and kernel pipelining co-design for asymmetric hardware scaling, 2026. URL https://arxiv.org/abs/2603.05451.

Krish Agarwal, Zhuoming Chen, Cheng Luo, Yongqi Chen, Haizhong Zheng, Xun Huang, Atri Rudra, and Beidi Chen. Monarchrt: Efficient attention for real-time video generation, 2026. URL https://arxiv.org/abs/2602.12271.

Siran Liu, Guoxia Wang, Sa Wang, Jinle Zeng, HaoYang Xie, Siyu Lou, JiaBin Yang, DianHai Yu, Haifeng Wang, and Chao Yang. Rrattention: Dynamic block sparse attention via per-head round-robin shifts for long-context inference, 2026. URL https://arxiv.org/abs/2602.05853.

David W Romero, Anna Kuzina, Erik J Bekkers, Jakub M Tomczak, and Mark Hoogendoorn. CKConv: Continuous kernel convolution for sequential data. *arXiv preprint arXiv:2102.02611*, 2021.

Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In *The International Conference on Machine Learning (ICML)*, pages 1120–1128, 2016.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/5dfbe6f5671e82c76841ba687a8a9ecb-Abstract-Conference.html.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. QUEST: query-aware sparsity for efficient long-context LLM inference. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, Proceedings of Machine Learning Research, pages 47901–47911. PMLR / OpenReview.net, 2024b. URL https://proceedings.mlr.press/v235/tang24l.html.

Ruyi Xu, Guangxuan Xiao, Haofeng Huang, Junxian Guo, and Song Han. Xattention: Block sparse attention with antidiagonal scoring. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, Proceedings of Machine Learning Research. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/xu25ag.html.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Zichuan Fu, Wentao Song, Yejing Wang, Xian Wu, Yefeng Zheng, Yingying Zhang, Derong Xu, Xuetao Wei, Tong Xu, and Xiangyu Zhao. Sliding window attention training for efficient large language models, 2025. URL https://arxiv.org/abs/2502.18845.

Aditya Desai, Shuo Yang, Alejandro Cuadron, Ana Klimovic, Matei Zaharia, Joseph E Gonzalez, and Ion Stoica. Hashattention: Semantic sparsity for faster inference. *arXiv preprint arXiv:2412.14468*, 2024.

Juechu Dong, Boyuan Feng, Driss Guessous, Yanbo Liang, and Horace He. Flex attention: A programming model for generating optimized attention kernels, 2024. URL https://arxiv.org/abs/2412.05496.

Zihao Ye, Lequn Chen, Ruihang Lai, Wuwei Lin, Yineng Zhang, Stephanie Wang, Tianqi Chen, Baris Kasikci, Vinod Grover, Arvind Krishnamurthy, and Luis Ceze. Flashinfer: Efficient and customizable attention engine for llm inference serving, 2025. URL https://arxiv.org/abs/2501.01005.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR, 2020.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *International Conference on Machine Learning*, 2024.

Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Zou, Atri Rudra, and Christopher Ré. Zoology: Measuring and improving recall in efficient language models. In *The International Conference on Learning Representations*, 2024.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024a.

Guangda Liu, Chengwei Li, Jieru Zhao, Chenqi Zhang, and Minyi Guo. Clusterkv: Manipulating llm kv cache in semantic space for recallable compression. *arXiv preprint arXiv:2412.03213*, 2024b.

Yuxiang Huang, Chaojun Xiao, Xu Han, and Zhiyuan Liu. Nosa: Native and offloadable sparse attention, 2025. URL https://arxiv.org/abs/2510.13602.

Meta NVIDIA. Pytorch container image. https://catalog.ngc.nvidia.com/orgs/nvidia/containers/pytorch, 2022.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. Gpt-neox-20b: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745*, 2022.

Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *ArXiv*, abs/1711.05101, 2017. URL https://api.semanticscholar.org/CorpusID:3312944.

Alex Hägele, Elie Bakouch, Atli Kosson, Leandro Von Werra, Martin Jaggi, et al. Scaling laws and compute-optimal training beyond fixed training durations. *Advances in Neural Information Processing Systems*, 37:76232–76264, 2024.

Houyi Li, Wenzhen Zheng, Qiufeng Wang, Hanshan Zhang, Zili Wang, Shijie Xuyang, Yuantao Fan, Shuigeng Zhou, Xiangyu Zhang, and Daxin Jiang. Predictable scale: Part i – optimal hyperparameter scaling law in large language model pretraining, 2025. URL https://arxiv.org/abs/2503.04715.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. An empirical analysis of compute-optimal large language model training. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:30016–30030, 2022.

Loubna Ben Allal, Anton Lozhkov, Guilherme Penedo, Thomas Wolf, and Leandro von Werra. Smollm-corpus, July 2024.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, Jason Phang, Laria Reynolds, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, September 2021. URL https://doi.org/10.5281/zenodo.5371628.

Clémentine Fourrier, Nathan Habib, Hynek Kydlíček, Thomas Wolf, and Lewis Tunstall. Lighteval: A lightweight framework for llm evaluation, 2023. URL https://github.com/huggingface/lighteval.

DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenhao Xu, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Erhang Li, Fangqi Zhou, Fangyun Lin, Fucong Dai, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Li, Haofen Liang, Haoran Wei, Haowei Zhang, Haowen Luo, Haozhe Ji, Honghui Ding, Hongxuan Tang, Huanqi Cao, Huazuo Gao, Hui Qu, Hui Zeng, Jialiang Huang, Jiashi Li, Jiaxin Xu, Jiewen Hu, Jingchang Chen, Jingting Xiang, Jingyang Yuan, Jingyuan Cheng, Jinhua Zhu, Jun Ran, Junguang Jiang, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Kexin Huang, Kexing Zhou, Kezhao Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Wang, Liang Zhao, Liangsheng Yin, Lihua Guo, Lingxiao Luo, Linwang Ma, Litong Wang, Liyue Zhang, M. S. Di, M. Y Xu, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Panpan Huang, Peixin Cong, Peiyi Wang, Qiancheng Wang, Qihao Zhu, Qingyang Li, Qinyu Chen, Qiushi Du, Ruiling Xu, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runqiu Yin, Runxin Xu, Ruomeng Shen, Ruoyu Zhang, S. H. Liu, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaofei Cai, Shaoyuan Chen, Shengding Hu, Shengyu Liu, Shiqiang Hu, Shirong Ma, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, Songyang Zhou, Tao Ni, Tao Yun, Tian Pei, Tian Ye, Tianyuan Yue, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjie Pang, Wenjing Luo, Wenjun Gao, Wentao Zhang, Xi Gao, Xiangwen Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaokang Zhang, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xingyou Li, Xinyu Yang, Xinyuan Li, Xu Chen, Xuecheng Su, Xuehai Pan, Xuheng Lin, Xuwei Fu, Y. Q. Wang, Yang Zhang, Yanhong Xu, Yanru Ma, Yao Li, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Qian, Yi Yu, Yichao Zhang, Yifan Ding, Yifan Shi, Yiliang Xiong, Ying He, Ying Zhou, Yinmin Zhong, Yishi Piao, Yisong Wang, Yixiao Chen, Yixuan Tan, Yixuan Wei, Yiyang Ma, Yiyuan Liu, Yonglun Yang, Yongqiang Guo, Yongtong Wu, Yu Wu, Yuan Cheng, Yuan Ou, Yuanfan Xu, Yuduan Wang, Yue Gong, Yuhan Wu, Yuheng Zou, Yukun Li, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehua Zhao, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhixian Huang, Zhiyu Wu, Zhuoshu Li, Zhuping Zhang, Zian Xu, Zihao Wang, Zihui Gu, Zijia Zhu, Zilin Li, Zipeng Zhang, Ziwei Xie, Ziyi Gao, Zizheng Pan, Zongqing Yao, Bei Feng, Hui Li, J. L. Cai, Jiaqi Ni, Lei Xu, Meng Li, Ning Tian, R. J. Chen, R. L. Jin, S. S. Li, Shuang Zhou, Tianyu Sun, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xinnan Song, Xinyi Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, Dongjie Ji, Jian Liang, Jianzhong Guo, Jin Chen, Leyi Xia, Miaojun Wang, Mingming Li, Peng Zhang, Ruyi Chen, Shangmian Sun, Shaoqing Wu, Shengfeng Ye, T. Wang, W. L. Xiao, Wei An, Xianzu Wang, Xiaowen Sun, Xiaoxiang Wang, Ying Tang, Yukun Zha, Zekai Zhang, Zhe Ju, Zhen Zhang, and Zihua Qu. Deepseek-v3.2: Pushing the frontier of open large language models, 2025. URL https://arxiv.org/abs/2512.02556.

Yizhao Gao, Jianyu Wei, Qihao Zhang, Yu Cheng, Shimao Chen, Zhengju Tang, Zihan Jiang, Yifan Song, Hailin Zhang, Liang Zhao, Bo Yang, Gang Wang, Shijie Cao, and Fuli Luo. Hysparse: A hybrid sparse attention architecture with oracle token selection and kv cache sharing, 2026. URL https://arxiv.org/abs/2602.03560.

Jiayi Yuan, Cameron Shinn, Kai Xu, Jingze Cui, George Klimiashvili, Guangxuan Xiao, Perkz Zheng, Bo Li, Yuxin Zhou, Zhouhai Ye, Weijie You, Tian Zheng, Dominic Brown, Pengbo Wang, Richard Cai, Julien Demouth, John D. Owens, Xia Hu, Song Han, Timmy Liu, and Huizi Mao. Blasst: Dynamic blocked attention sparsity via softmax thresholding, 2025b. URL https://arxiv.org/abs/2512.12087.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc-Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1525–1534, 2016.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension, 2017.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on Artificial Intelligence*, volume 34, 2020.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial Winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051, 2023.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*, 2023.

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.

G Kamradt. Llmtest needleinahaystack, 2023. URL `https://github.com/gkamradt/LLMTest_NeedleInAHaystack`.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

AI-MO. Aime 2024, 2024. URL `https://huggingface.co/datasets/AI-MO/aimo-validation-aime`.

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

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

Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

A Proof of Block-Level Rank-1 Approximation

This section formalizes the modeling claim behind the rank-1 gate used in DMA. We do *not* assume that a head is literally rank-1 at the token level, nor that a single direction explains every sample and every position exactly. The statement proved below is narrower and more relevant to the kernel design: if the *block-averaged* query and key statistics of one head are dominated by one query direction and one key direction, then the corresponding block-level attention matrix admits a controlled rank-1 approximation. This is precisely the regime in which the lightweight factors used by DMA are expected to serve as an effective coarse block selector.

Block means and averaged attention. Fix one attention head with query and key vectors $q_i, k_j \in \mathbb{R}^{d_h}$. Let $\{I_a\}_{a=1}^{T_r}$ be a partition of the query indices and $\{J_b\}_{b=1}^{T_c}$ be a partition of the key indices. Define the block means

$$\bar{q}_a = \frac{1}{|I_a|} \sum_{i \in I_a} q_i, \qquad \bar{k}_b = \frac{1}{|J_b|} \sum_{j \in J_b} k_j, \tag{11}$$

and the block-averaged attention matrix $M \in \mathbb{R}^{T_r \times T_c}$ by

$$M_{ab} = \frac{1}{|I_a||J_b|} \sum_{i \in I_a} \sum_{j \in J_b} q_i^\top k_j. \tag{12}$$

Block averaging identity. For every query block a and key block b,

$$M_{ab} = \bar{q}_a^\top \bar{k}_b. \tag{13}$$

Equivalently, if $\bar{Q} \in \mathbb{R}^{T_r \times d_h}$ and $\bar{K} \in \mathbb{R}^{T_c \times d_h}$ collect the block means row-wise, then

$$M = \bar{Q}\bar{K}^\top. \tag{14}$$

Proof. By bilinearity of the inner product and separability of the sums,

$$\begin{aligned} M_{ab} &= \frac{1}{|I_a||J_b|} \sum_{i \in I_a} \sum_{j \in J_b} q_i^\top k_j \\ &= \left(\frac{1}{|I_a|} \sum_{i \in I_a} q_i \right)^\top \left(\frac{1}{|J_b|} \sum_{j \in J_b} k_j \right) = \bar{q}_a^\top \bar{k}_b. \end{aligned} \tag{15}$$

Stacking all block means row-wise yields $M = \bar{Q}\bar{K}^\top$. □

The block averaging identity above shows that block averaging does not destroy the bilinear structure of attention. It only changes the operands from token features to block means. Thus, any low-rank claim made at the block level concerns the matrix $M = \bar{Q}\bar{K}^\top$, not the full token-level matrix $QK^\top$.

Dominant-direction hypothesis. Assume there exist unit vectors $u, v \in \mathbb{R}^{d_h}$ such that each block mean decomposes as

$$\bar{q}_a = \alpha_a u + r_a, \qquad \bar{k}_b = \delta_b v + s_b, \tag{16}$$

where

$$\alpha_a = u^\top \bar{q}_a, \qquad \delta_b = v^\top \bar{k}_b, \qquad u^\top r_a = 0, \qquad v^\top s_b = 0. \tag{17}$$

Here α_a and δ_b capture the dominant scalar coordinate of the block mean along the query and key directions, while r_a and s_b are the orthogonal residuals.

Define the coefficient vectors $\alpha \in \mathbb{R}^{T_r}$ and $\delta \in \mathbb{R}^{T_c}$, and let $R \in \mathbb{R}^{T_r \times d_h}$ and $S \in \mathbb{R}^{T_c \times d_h}$ collect the residuals row-wise, so that

$$\bar{Q} = \alpha u^\top + R, \qquad \bar{K} = \delta v^\top + S. \tag{18}$$

The natural rank-1 approximation induced by the dominant directions is

$$\widehat{M} = (u^\top v)\, \alpha \delta^\top, \qquad \widehat{M}_{ab} = (u^\top v)\, \alpha_a \delta_b. \tag{19}$$

Rank-1 approximation error decomposition. Under the decomposition in Equations equation 16–equation 19, the block attention matrix satisfies the exact error decomposition

$$M - \widehat{M} = \alpha(Su)^\top + (Rv)\delta^\top + RS^\top, \tag{20}$$

and, equivalently, for each block pair (a, b),

$$M_{ab} - \widehat{M}_{ab} = \alpha_a u^\top s_b + \delta_b r_a^\top v + r_a^\top s_b. \tag{21}$$

Therefore the entrywise approximation error obeys

$$|M_{ab} - \widehat{M}_{ab}| \le |\alpha_a| \, \|s_b\|_2 + |\delta_b| \, \|r_a\|_2 + \|r_a\|_2 \, \|s_b\|_2. \tag{22}$$

Moreover, if

$$A = \max_a |\alpha_a|, \qquad D = \max_b |\delta_b|, \qquad R_q = \max_a \|r_a\|_2, \qquad R_k = \max_b \|s_b\|_2, \tag{23}$$

then

$$\|M - \widehat{M}\|_{\max} \le AR_k + DR_q + R_q R_k. \tag{24}$$

If, in addition, $\|R\|_F^2 = \sum_a \|r_a\|_2^2$ and $\|S\|_F^2 = \sum_b \|s_b\|_2^2$, then

$$\|M - \widehat{M}\|_F \le \|\alpha\|_2 \, \|S\|_F + \|\delta\|_2 \, \|R\|_F + \|R\|_F \, \|S\|_F. \tag{25}$$

Proof. Substituting Equation equation 18 into Equation equation 14 gives

$$\begin{aligned} M &= \bar{Q}\bar{K}^\top \\ &= (\alpha u^\top + R)(\delta v^\top + S)^\top \\ &= (\alpha u^\top + R)(v\delta^\top + S^\top) \\ &= (u^\top v)\, \alpha\delta^\top + \alpha(u^\top S^\top) + (Rv)\delta^\top + RS^\top. \end{aligned} \tag{26}$$

Because $u^\top S^\top = (Su)^\top$, subtracting $\widehat{M} = (u^\top v)\alpha\delta^\top$ from Equation equation 26 yields Equation equation 20. Entrywise, the a-th component of Rv is $r_a^\top v$, so Equation equation 21 follows immediately:

$$\begin{aligned} M_{ab} &= (\alpha_a u + r_a)^\top (\delta_b v + s_b) \\ &= (u^\top v)\, \alpha_a \delta_b + \alpha_a u^\top s_b + \delta_b r_a^\top v + r_a^\top s_b. \end{aligned} \tag{27}$$

Subtracting $\widehat{M}_{ab}$ yields Equation equation 21. Applying Cauchy–Schwarz gives

$$|u^\top s_b| \le \|u\|_2 \|s_b\|_2 = \|s_b\|_2, \qquad |r_a^\top v| \le \|r_a\|_2 \|v\|_2 = \|r_a\|_2, \qquad |r_a^\top s_b| \le \|r_a\|_2 \|s_b\|_2, \tag{28}$$

which proves Equation equation 22. Taking the supremum over (a, b) gives Equation equation 24. For the Frobenius bound, the decomposition

$$M - \widehat{M} = \alpha(Su)^\top + (Rv)\delta^\top + RS^\top \tag{29}$$

and the triangle inequality imply

$$\|M - \widehat{M}\|_F \le \|\alpha(Su)^\top\|_F + \|(Rv)\delta^\top\|_F + \|RS^\top\|_F. \tag{30}$$

Using $\|xy^\top\|_F = \|x\|_2 \|y\|_2$, $\|Su\|_2 \le \|S\|_F \|u\|_2 = \|S\|_F$, $\|Rv\|_2 \le \|R\|_F$, and $\|RS^\top\|_F \le \|R\|_F \|S\|_F$ yields Equation equation 25. □

The error decomposition above states the precise condition under which DMA's rank-1 gate is justified at block level: the block means must be well aligned with one dominant query direction and one dominant key direction, so that the orthogonal residuals are small. The approximation error is then controlled by the residual magnitudes and by the strength of the leading coefficients α_a, δ_b.

Connection to learned scalar gates. Consider learned projection vectors $w_A, w_\Delta \in \mathbb{R}^{d_h}$ and define token-wise gate factors by

$$A_i = q_i^\top w_A, \qquad \Delta_j = k_j^\top w_\Delta. \tag{31}$$

Their block means are exactly

$$\bar{A}_a = \frac{1}{|I_a|}\sum_{i\in I_a} A_i = w_A^\top \bar{q}_a, \qquad \bar{\Delta}_b = \frac{1}{|J_b|}\sum_{j\in J_b} \Delta_j = w_\Delta^\top \bar{k}_b. \tag{32}$$

Now decompose the learned directions relative to the dominant directions as

$$w_A = c_A u + e_A, \qquad w_\Delta = c_\Delta v + e_\Delta, \qquad u^\top e_A = 0, \qquad v^\top e_\Delta = 0. \tag{33}$$

Then, using Equation equation 16,

$$\bar{A}_a = c_A \alpha_a + e_A^\top r_a, \qquad \bar{\Delta}_b = c_\Delta \delta_b + e_\Delta^\top s_b. \tag{34}$$

Hence the block-level rank-1 gate satisfies

$$\bar{A}_a\bar{\Delta}_b - c_A c_\Delta\, \alpha_a\delta_b = c_A\alpha_a e_\Delta^\top s_b + c_\Delta\delta_b e_A^\top r_a + (e_A^\top r_a)(e_\Delta^\top s_b), \tag{35}$$

and therefore

$$\left|\bar{A}_a\bar{\Delta}_b - c_A c_\Delta\, \alpha_a\delta_b\right| \le |c_A|\,|\alpha_a|\,\|e_\Delta\|_2\,\|s_b\|_2 + |c_\Delta|\,|\delta_b|\,\|e_A\|_2\,\|r_a\|_2 + \|e_A\|_2\,\|e_\Delta\|_2\,\|r_a\|_2\,\|s_b\|_2. \tag{36}$$

This shows that the learned scalar gate used by DMA is close to the dominant rank-1 structure when two conditions hold simultaneously: the learned projections align with the dominant directions, and the block residuals remain small.

Spectral interpretation. Let M_1 be the best rank-1 approximation of M in Frobenius norm. Because $\widehat{M}$ in Equation equation 19 is itself rank-1, the Eckart–Young theorem implies

$$\|M - M_1\|_F = \min_{\mathrm{rank}(X)\le 1} \|M - X\|_F \le \|M - \widehat{M}\|_F. \tag{37}$$

If $\sigma_1(M) \ge \sigma_2(M) \ge \cdots \ge 0$ are the singular values of M, then

$$\sum_{t\ge 2}\sigma_t(M)^2 = \|M - M_1\|_F^2 \le \left(\|\alpha\|_2\,\|S\|_F + \|\delta\|_2\,\|R\|_F + \|R\|_F\,\|S\|_F\right)^2. \tag{38}$$

Thus, small dominant-direction residuals imply that most of the block-level energy of M is concentrated in its leading singular mode. This is the precise matrix-approximation statement behind the heuristic phrase "dominant direction" used in the main text.

Implication for DMA. The result above is only meant to justify the gate as a block-level screening signal. DMA does not assume that token-wise attention is globally low-rank. The rank-1 form is used only to estimate which $B \times B$ tiles should be computed; once a tile is retained, attention within that tile is evaluated with the full token-wise scores. In other words, the theorem supports a cheap predictor of coarse block salience, rather than a low-rank substitute for attention itself. If some heads are governed by several comparable block-level modes rather than a single dominant one, a more expressive screen would likely be built from richer blockwise QK statistics rather than by stacking many scalar-factor pairs, since the latter increases predictor cost while eroding the simplicity of the current kernel path. Such a screen could still prune tiles before paying the full softmax and PV cost, but we restrict attention to the rank-1 case in this work.

B ALGORITHMS FOR DYNAMIC MASK ATTENTION

Overview. This section provides the complete pseudocode for the forward Algorithm 1 and backward Algorithm 2 passes of DMA. The algorithms formalize the loop orderings, the blockwise screening mechanisms, and the state retained in fast memory during execution. They also make explicit where DMA departs from dense FlashAttention: the dense tile loop is preceded by a learned block-level gate test, while all retained tiles still use the usual token-wise score computation and online softmax accumulation. The discussion below is organized around the forward pass and backward pass separately, so that the execution-time screening rule, retained-tile computation, and approximate training dynamics can be read directly against the corresponding pseudocode.

Forward pass. The forward pass employs a query-major traversal consistent with FlashAttention-style online softmax accumulation. For each active query block, the kernel scans candidate key blocks, maintains the running reference φ_i, and updates the online softmax state only for retained tiles. Screening is performed before loading the full key and value tile, using the scalar factors, block extrema, upper-bound test, and position-dependent margin. This makes the sparse execution decision part of the tiled attention loop itself rather than a separate preprocessing stage.

Threshold schedule. The forward pass applies the blockwise upper bound Equation equation 3 and the position-dependent margin threshold Equation equation 5. The threshold is a relative improvement margin, not an absolute cutoff on either attention logits or gate values. A candidate tile is retained only when its blockwise gate upper bound improves over the current running reference by at least θ_i; otherwise, the tile is skipped before the full key and value blocks are loaded. The position factor in θ_i compensates for the causal growth of the accessible prefix: early query blocks have few candidate keys and therefore use a mild margin, while later query blocks face a larger candidate set and can tolerate stricter pruning. This is the mechanism behind the dynamic sparsity schedule used in the experiments: τ sets the base quality-efficiency trade-off, while the query position modulates how aggressively the kernel prunes as more keys become available.

Cost of the screening path. The screening condition depends only on the scalar factors $\mathbf{A}$, $\mathbf{\Delta}$ and their per-block extrema. Once these extrema are available, evaluating a candidate tile requires only four scalar products, a maximum over those products, and one comparison against the running reference and margin. The key and value tiles are loaded from HBM only after the block clears this test. Thus the screening path is effectively constant overhead per candidate tile and is small compared with the work avoided on skipped tiles: loading $\mathbf{K}_j$ and $\mathbf{V}_j$, forming the full $\mathbf{Q}_i\mathbf{K}_j^\top$ score matrix, applying softmax updates, and multiplying by $\mathbf{V}_j$. This is why DMA places the gate test before the expensive memory movement rather than after materializing the full attention tile.

Retained-tile computation. For every retained tile, the same gate is used as a continuous additive logit bias rather than as a replacement for token-wise attention. Since $\log \sigma(\cdot) \leq 0$, the gate only suppresses unnormalized attention mass, equivalently multiplying the exponentiated attention weights by factors in $(0, 1)$. This gives the gate a smooth multiplicative interpretation without introducing hard routing inside a retained tile. The retained score matrix is still $\mathbf{Q}_i\mathbf{K}_j^\top + \mathbf{G}_{ij}$, so the full token-level $\mathbf{Q}_i\mathbf{K}_j^\top$ computation is preserved wherever a tile is materialized. In implementation, forming $\mathbf{G}_{ij}$ amounts to an outer product of the scalar factors followed by the pointwise $\log \sigma(\cdot)$ transform and a fused addition into the score matrix. This extra arithmetic is small relative to the matrix products and online softmax-value accumulation already performed for the retained tile.

Backward pass. The backward pass uses a key-major traversal to facilitate accumulation of $\mathbf{dK}$, $\mathbf{dV}$, and $\mathbf{d\Delta}$ in registers before writing them back to HBM. This change in loop order is the reason the backward kernel maintains an analogous running reference φ_j for the active key block. Screening is reconstructed from the same learned scalar factors, block extrema, upper-bound test, and position-dependent margin, rather than from a stored binary routing mask. The resulting sparse pattern is therefore traversal-dependent, but it avoids storing forward routing decisions and keeps gradient accumulation aligned with the memory layout of the key and value blocks.

Online reconstruction. The backward pass applies the same kind of gate-derived upper-bound test while reconstructing sparsity online. It should not be interpreted as replaying the full dense

Algorithm 1 Dynamic Masked Attention Forward

Require: Matrices $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N \times d_h}$, Vectors $\mathbf{A}, \mathbf{\Delta} \in \mathbb{R}^N$ in HBM. Block sizes B, Threshold τ.

Initialize $\mathbf{O} = (0) \in \mathbb{R}^{N \times d_h}, \ell = (0) \in \mathbb{R}^N$ in HBM.
Divide $\mathbf{Q}, \mathbf{O}$ into T_r blocks of size $B \times d_h$, and divide $\mathbf{A}, \ell$ into T_r blocks of size B.
Divide $\mathbf{K}, \mathbf{V}$ into T_c blocks of size $B \times d_h$, and divide $\mathbf{\Delta}$ into T_c blocks of size B.
for $1 \le i \le T_r$ **do**
  Initialize $\mathbf{O}_i = (0) \in \mathbb{R}^{B \times d_h}, \varphi_i = (-\infty) \in \mathbb{R}, \ell_i = (0) \in \mathbb{R}^B, m_i = (-\infty) \in \mathbb{R}^B$ in REG.
  Load $\mathbf{Q}_i, \mathbf{A}_i$ from HBM to SRAM.
  Compute $\theta_i = \tau \cdot \frac{iB}{N}$.
  for $T_c \ge j \ge 1$ **do**
    Load $\mathbf{\Delta}_j$ from HBM to SRAM.
    Compute $\tilde{\varphi}_{ij} = \max(\mathbf{A}_i^{\max}\mathbf{\Delta}_j^{\max},\ \mathbf{A}_i^{\max}\mathbf{\Delta}_j^{\min},\ \mathbf{A}_i^{\min}\mathbf{\Delta}_j^{\max},\ \mathbf{A}_i^{\min}\mathbf{\Delta}_j^{\min})$.
    if $\tilde{\varphi}_{ij} - \varphi_i < \theta_i$ **then**
      skip block (i, j) and **continue**.
    end if
    Compute $\varphi_i^{\text{new}} = \max(\varphi_i, \tilde{\varphi}_{ij})$.
    Update $\varphi_i \leftarrow \varphi_i^{\text{new}}$ in REG.
    Compute $\mathbf{G}_{ij} = \log \sigma(\mathbf{A}_i \mathbf{\Delta}_j^T) \in \mathbb{R}^{B \times B}$.
    Load $\mathbf{K}_j, \mathbf{V}_j$ from HBM to SRAM.
    Compute $\mathbf{S}_{ij} = \mathbf{Q}_i \mathbf{K}_j^T + \mathbf{G}_{ij} \in \mathbb{R}^{B \times B}$.
    Compute $\tilde{m}_{ij} = \text{rowmax}(\mathbf{S}_{ij}) \in \mathbb{R}^B$.
    Compute $\tilde{\mathbf{P}}_{ij} = \exp(\mathbf{S}_{ij} - \tilde{m}_{ij}) \in \mathbb{R}^{B \times B}, \tilde{\ell}_{ij} = \text{rowsum}(\tilde{\mathbf{P}}_{ij}) \in \mathbb{R}^B$.
    Compute $m_i^{\text{new}} = \max(m_i, \tilde{m}_{ij}) \in \mathbb{R}^B, \ell_i^{\text{new}} = e^{m_i - m_i^{\text{new}}} \ell_i + e^{\tilde{m}_{ij} - m_i^{\text{new}}} \tilde{\ell}_{ij} \in \mathbb{R}^B$.
    Update $\mathbf{O}_i \leftarrow e^{m_i - m_i^{\text{new}}} \mathbf{O}_i + e^{\tilde{m}_{ij} - m_i^{\text{new}}} \tilde{\mathbf{P}}_{ij} \mathbf{V}_j$ in REG.
    Update $\ell_i \leftarrow \ell_i^{\text{new}},\ m_i \leftarrow m_i^{\text{new}}$ in REG.
  end for
  Update $\mathbf{O}_i \leftarrow \text{diag}(\ell_i)^{-1} \mathbf{O}_i$ in REG.
  Update $\ell_i \leftarrow m_i + \log(\ell_i)$ in REG.
  Write $\mathbf{O}_i, \ell_i$ to HBM.
end for
Return $\mathbf{O}, \ell$.

computation or as proving that the backward traversal selects exactly the same sparse set as the forward traversal. Rather, the backward kernel uses the same learned scalar factors and threshold rule under a loop order chosen for efficient gradient accumulation. Blocks that pass this online test are materialized and differentiated normally; blocks that fail it are skipped without forming their scores, probabilities, or gate values.

Gradients on retained tiles. For a retained tile, the backward path is fully differentiable. The kernel reconstructs the retained scores and probabilities from the stored log-partition statistic ℓ_i, computes the standard score gradient $\mathbf{dS}_{ij}$, and then propagates it both through the ordinary attention path and through the gate path. The ordinary path contributes to $\mathbf{dQ}_i$, $\mathbf{dK}_j$, and $\mathbf{dV}_j$ using the same algebra as FlashAttention. The gate path uses the derivative of $\log \sigma(\mathbf{A}_i \mathbf{\Delta}_j^\top)$, producing $\mathbf{dG}_{ij} = \mathbf{dS}_{ij} \circ \sigma(-\mathbf{A}_i \mathbf{\Delta}_j^\top)$ and then accumulating $\mathbf{dA}_i$ and $\mathbf{d\Delta}_j$ through the rank-1 bilinear structure. These scalar-factor gradients subsequently backpropagate through the projection directions w_A and w_Δ and into the query and key representations that produced the factors. Thus, whenever DMA retains a tile, any mismatch between the learned gate and the downstream loss produces a direct corrective signal for the predictor.

What happens to skipped tiles. A skipped tile has no direct contribution to $\mathbf{dQ}$, $\mathbf{dK}$, $\mathbf{dV}$, $\mathbf{dA}$, or $\mathbf{d\Delta}$ for that particular block pair because the corresponding $\mathbf{S}_{ij}$, $\mathbf{G}_{ij}$, softmax probabilities, and value product are never materialized. This is the main approximation made by DMA during training: the skipped interaction is treated as zero work and receives no direct gradient for that query-key block. The approximation is nevertheless controlled by the screening rule. A block is skipped only when its gate upper bound fails to clear the traversal-specific improvement margin, and the forward and backward error analysis in Section C shows how omitted contributions are governed by the skipped dense probability mass, together with corrections from truncated-softmax renormalization and dense-versus-truncated output mismatch.

Algorithm 2 Dynamic Masked Attention Backward

Require: Matrices $\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{O}, \mathbf{dO} \in \mathbb{R}^{N \times d_h}$, Vectors $\mathbf{A}, \mathbf{\Delta}, \ell \in \mathbb{R}^N$ in HBM. Block sizes B, Threshold τ.

Initialize $\mathbf{dQ} = (0) \in \mathbb{R}^{N \times d_h}, \mathbf{dK} = (0) \in \mathbb{R}^{N \times d_h}, \mathbf{dV} = (0) \in \mathbb{R}^{N \times d_h}$ in HBM.
Initialize $\mathbf{dA} = (0) \in \mathbb{R}^N, \mathbf{d\Delta} = (0) \in \mathbb{R}^N$ in HBM.
Divide $\mathbf{Q}, \mathbf{dQ}$ into T_r blocks of size $B \times d_h$, and divide $\mathbf{A}, \ell, \mathbf{dA}$ into T_r blocks of size B.
Divide $\mathbf{K}, \mathbf{V}, \mathbf{dK}, \mathbf{dV}$ into T_c blocks of size $B \times d_h$, and divide $\mathbf{\Delta}, \mathbf{d\Delta}$ into T_c blocks of size B.
for $1 \le j \le T_c$ **do**
  Initialize $\mathbf{dK}_j = (0) \in \mathbb{R}^{B \times d_h}, \mathbf{dV}_j = (0) \in \mathbb{R}^{B \times d_h}, \mathbf{d\Delta}_j = (0) \in \mathbb{R}^B, \varphi_j = (-\infty) \in \mathbb{R}$ in REG.
  Load $\mathbf{K}_j, \mathbf{V}_j, \mathbf{\Delta}_j$ from HBM to SRAM.
  for $1 \le i \le T_r$ **do**
    Load $\mathbf{Q}_i, \mathbf{A}_i, \mathbf{O}_i, \mathbf{dO}_i, \ell_i$ from HBM to SRAM.
    Compute $\theta_i = \tau \cdot \frac{iB}{N}$.
    Compute $\tilde{\varphi}_{ij} = \max(\mathbf{A}_i^{\max}\mathbf{\Delta}_j^{\max},\ \mathbf{A}_i^{\max}\mathbf{\Delta}_j^{\min},\ \mathbf{A}_i^{\min}\mathbf{\Delta}_j^{\max},\ \mathbf{A}_i^{\min}\mathbf{\Delta}_j^{\min})$.
    if $\tilde{\varphi}_{ij} - \varphi_j < \theta_i$ **then**
      skip block (i, j) and **continue**.
    end if
    Compute $\varphi_j^{\text{new}} = \max(\varphi_j, \tilde{\varphi}_{ij})$.
    Update $\varphi_j \leftarrow \varphi_j^{\text{new}}$ in REG.
    Compute $\mathbf{G}_{ij} = \log \sigma(\mathbf{A}_i \mathbf{\Delta}_j^T) \in \mathbb{R}^{B \times B}$.
    Compute $\mathbf{S}_{ij} = \mathbf{Q}_i \mathbf{K}_j^T + \mathbf{G}_{ij} \in \mathbb{R}^{B \times B}$.
    Compute $\mathbf{P}_{ij} = \exp(\mathbf{S}_{ij} - \ell_i) \in \mathbb{R}^{B \times B}$.
    Update $\mathbf{dV}_j \leftarrow \mathbf{dV}_j + \mathbf{P}_{ij}^T \mathbf{dO}_i$ in REG.
    Compute $\mathbf{dP}_{ij} = \mathbf{dO}_i \mathbf{V}_j^T \in \mathbb{R}^{B \times B}$.
    Compute $\mathbf{dS}_{ij} = \mathbf{P}_{ij} \circ (\mathbf{dP}_{ij} - \text{rowsum}(\mathbf{dO}_i \circ \mathbf{O}_i)) \in \mathbb{R}^{B \times B}$.
    Update $\mathbf{dQ}_i \leftarrow \mathbf{dQ}_i + \mathbf{dS}_{ij} \mathbf{K}_j$ in HBM.
    Update $\mathbf{dK}_j \leftarrow \mathbf{dK}_j + \mathbf{dS}_{ij}^T \mathbf{Q}_i$ in REG.
    Compute $\mathbf{dG}_{ij} = \mathbf{dS}_{ij} \circ \sigma(-\mathbf{A}_i \mathbf{\Delta}_j^T) \in \mathbb{R}^{B \times B}$.
    Update $\mathbf{dA}_i \leftarrow \mathbf{dA}_i + \mathbf{dG}_{ij} \mathbf{\Delta}_j$ in HBM.
    Update $\mathbf{d\Delta}_j \leftarrow \mathbf{d\Delta}_j + \mathbf{dG}_{ij}^T \mathbf{A}_i$ in REG.
  end for
  Write $\mathbf{dK}_j, \mathbf{dV}_j, \mathbf{d\Delta}_j$ to HBM.
end for
Return $\mathbf{dQ}, \mathbf{dK}, \mathbf{dV}, \mathbf{dA}, \mathbf{d\Delta}$.

Why the predictor can still learn. The absence of a direct gradient on one skipped block does not make the gate an external heuristic. The scalar factors $\mathbf{A}$ and $\mathbf{\Delta}$ are shared across many query-key interactions in the same sequence and across many examples during training. Retained tiles therefore provide many supervised paths through which the projection directions w_A and w_Δ are updated. If the current gate is too aggressive for a recurring interaction pattern, related interactions that remain retained can increase the corresponding scalar factors, raising future block upper bounds and making similar tiles more likely to clear the margin. Conversely, interactions that consistently fail the improvement test remain in the low-priority tail under the learned gate and calibrated threshold. In this sense, DMA learns execution-time sparsity from the same downstream objective as attention on the retained computation path, rather than from a separate sparsity loss, a hand-designed cache heuristic, or a non-differentiable top-k router.

Traversal-dependent approximation. Because the backward loop is key-major while the forward loop is query-major, the running reference in backward is φ_j rather than φ_i. This makes the backward screening pattern traversal-dependent. The design choice is intentional: it avoids storing an explicit forward binary mask and allows $\mathbf{dK}$, $\mathbf{dV}$, and $\mathbf{d\Delta}$ to be accumulated efficiently for each key block. The cost is that backward sparsity should be viewed as a reconstruction from the same gate statistics under a different accumulation order, not as an exact replay of every forward routing decision. Section C therefore analyzes a reconstruction-based approximation rather than claiming equality with dense-attention gradients on all skipped interactions.

C Forward and Backward Skip Error Analysis

Forward skip error under the pseudocode. We now make explicit the approximation introduced by the skip branch in Algorithm 1. Fix a query block i, a query token $u \in I_i$, and a key block j that is skipped by Equation equation 6. Let

$$s_{uv} = q_u^\top k_v + g_{uv}, \qquad g_{uv} = \log \sigma(A_u \Delta_v), \qquad p^\star_{uv} = \frac{\exp(s_{uv})}{Z_u}, \tag{39}$$

for v in the causal set of query u, where $Z_u = \sum_t \exp(s_{ut})$ is the exact row partition function. In the executed DMA forward pass, every skipped block contributes zero, so the probability mass omitted by skipping block j is exactly

$$\varepsilon^{\mathrm{fwd}}_{u,j} = \sum_{v \in J_j} p^\star_{uv}. \tag{40}$$

Define the best materialized score already seen for row u before testing block (i, j) by

$$m_i(u) = \max\{s_{ut} : t \text{ lies in a block visited before } (i, j) \text{ and not skipped}\}, \tag{41}$$

and define the corresponding skip margin

$$\Gamma^{\mathrm{fwd}}_{u,j} = m_i(u) - \max_{v \in J_j} \Big(q_u^\top k_v + \log \sigma(\tilde{\varphi}_{ij}) \Big). \tag{42}$$

Then the skipped probability mass satisfies

$$\varepsilon^{\mathrm{fwd}}_{u,j} \le |J_j| \exp\big(- \Gamma^{\mathrm{fwd}}_{u,j} \big). \tag{43}$$

If, in addition, $\|v_t\|_2 \le V_{\max}$ for every value vector in the skipped block, then the corresponding output truncation error obeys

$$\left\| \sum_{v \in J_j} p^\star_{uv} v_v \right\|_2 \le V_{\max} |J_j| \exp\big(- \Gamma^{\mathrm{fwd}}_{u,j} \big). \tag{44}$$

Proof. Equation equation 4 implies that for every $v \in J_j$,

$$g_{uv} \le \log \sigma(\tilde{\varphi}_{ij}), \qquad s_{uv} \le q_u^\top k_v + \log \sigma(\tilde{\varphi}_{ij}) \le m_i(u) - \Gamma^{\mathrm{fwd}}_{u,j}. \tag{45}$$

Since $Z_u \ge \exp(m_i(u))$, we obtain

$$p^\star_{uv} = \frac{\exp(s_{uv})}{Z_u} \le \exp\big(s_{uv} - m_i(u)\big) \le \exp\big(- \Gamma^{\mathrm{fwd}}_{u,j} \big). \tag{46}$$

Summing over $v \in J_j$ yields Equation equation 43. The output bound follows from the triangle inequality:

$$\left\| \sum_{v \in J_j} p^\star_{uv} v_v \right\|_2 \le \sum_{v \in J_j} p^\star_{uv} \|v_v\|_2 \le V_{\max} \varepsilon^{\mathrm{fwd}}_{u,j}. \tag{47}$$

Substituting Equation equation 43 proves Equation equation 44. □

The forward bound above clarifies the approximation made by the executed forward pass: once a block is skipped, its contribution is set to zero rather than retained as a very small softmax weight. The resulting truncation error is small whenever the best materialized score $m_i(u)$ remains well above the largest score the skipped block could attain under the gate upper bound. Accordingly, the guarantee is expressed in terms of the total-score margin $\Gamma^{\mathrm{fwd}}_{u,j}$. Equation equation 6 controls the gate-side upper bound entering this margin, but it should not be read as a direct upper bound on the full score $q_u^\top k_v + g_{uv}$ by itself.

Backward skipped-block error under hypothetical reconstruction from forward statistics. Now consider Algorithm 2. For a fixed query token $u \in I_i$, let $\mathcal{M}_u$ denote the materialized key indices in Algorithm 1, let $\mathcal{S}_u$ denote the skipped ones, and define the truncated partition actually stored by the forward pseudocode as

$$\widehat{Z}_u = \sum_{t \in \mathcal{M}_u} \exp(s_{ut}), \qquad \widehat{\ell}_u = \log \widehat{Z}_u. \tag{48}$$

The exact dense partition is then

$$Z_u = \widehat{Z}_u + \sum_{t \in \mathcal{S}_u} \exp(s_{ut}), \tag{49}$$

and the total skipped dense probability mass on that row is

$$\delta_u = \sum_{t \in \mathcal{S}_u} p^\star_{ut}. \tag{50}$$

Equivalently,

$$\widehat{Z}_u = Z_u(1 - \delta_u). \tag{51}$$

For the same skipped block (i, j), suppose one ignores the skip branch in Algorithm 2 and instead reconstructs that block from the stored forward statistic $\widehat{\ell}_u$ as

$$\widehat{p}_{uv} = \exp(s_{uv} - \widehat{\ell}_u), \qquad v \in J_j, \tag{52}$$

Then

$$\sum_{v \in J_j} \widehat{p}_{uv} = \frac{\varepsilon^{\mathrm{fwd}}_{u,j}}{1 - \delta_u} \le \frac{|J_j| \exp\big(-\Gamma^{\mathrm{fwd}}_{u,j}\big)}{1 - \delta_u}. \tag{53}$$

Thus the reconstructed gradient contribution associated with the skipped block is governed by the same forward omitted mass, up to the renormalization factor $(1 - \delta_u)^{-1}$ induced by the truncated partition written by Algorithm 1. This should be interpreted as a reconstruction-based analysis derived from the stored forward statistics, rather than as a proof that the executed backward screening path selects exactly the same skipped set, or even a subset of it.

To compare the reconstructed quantities with their dense counterparts, define the dense and truncated outputs

$$o^\star_u = \sum_t p^\star_{ut} v_t, \qquad \widehat{o}_u = \sum_{t \in \mathcal{M}_u} \widehat{p}_{ut} v_t. \tag{54}$$

The next bound shows that the output mismatch is itself controlled by the skipped row mass.

Output mismatch bound. If $\|v_t\|_2 \le V_{\max}$ for all terms in row u, then

$$\|o^\star_u - \widehat{o}_u\|_2 \le 2V_{\max}\delta_u. \tag{55}$$

Proof. Using Equation equation 68 on the materialized set,

$$o^\star_u - \widehat{o}_u = \sum_{t \in \mathcal{M}_u} (p^\star_{ut} - \widehat{p}_{ut}) v_t + \sum_{t \in \mathcal{S}_u} p^\star_{ut} v_t. \tag{56}$$

Therefore,

$$\begin{aligned} \|o^\star_u - \widehat{o}_u\|_2 &\le \sum_{t \in \mathcal{M}_u} |p^\star_{ut} - \widehat{p}_{ut}| \, \|v_t\|_2 + \sum_{t \in \mathcal{S}_u} p^\star_{ut} \, \|v_t\|_2 \\ &\le V_{\max} \sum_{t \in \mathcal{M}_u} \frac{\delta_u}{1 - \delta_u} p^\star_{ut} + V_{\max} \sum_{t \in \mathcal{S}_u} p^\star_{ut} \\ &= V_{\max}\delta_u + V_{\max}\delta_u = 2V_{\max}\delta_u, \end{aligned} \tag{57}$$

because $\sum_{t \in \mathcal{M}_u} p^\star_{ut} = 1 - \delta_u$. □

More concretely, assume there exists $\rho \in [0, 1)$ such that

$$\delta_u \le \rho < 1 \qquad \text{for all query rows under consideration,} \tag{58}$$

and that the row-wise norms satisfy

$$\|\mathrm{d}o_u\|_2 \le G_O, \qquad \|o_u^\star\|_2 \le O_{\max}, \qquad \|\widehat{o}_u\|_2 \le O_{\max}, \qquad \|v_v\|_2 \le V_{\max}, \qquad \|k_v\|_2 \le K_{\max}, \tag{59}$$

for $u \in I_i$, $v \in J_j$, and define

$$C_{\mathrm{score}} = G_O(V_{\max} + O_{\max}). \tag{60}$$

If $\widehat{\mathrm{d}S}_{uv}$ denotes the score gradient that would be obtained from the non-skip branch of Algorithm 2, and if

$$\mathrm{d}S^\star_{uv} = p^\star_{uv}\big(\mathrm{d}o_u^\top v_v - \mathrm{d}o_u^\top o^\star_u\big) \tag{61}$$

denotes the corresponding dense score gradient, then

$$|\widehat{\mathrm{d}S}_{uv}| \le C_{\mathrm{score}}\widehat{p}_{uv}. \tag{62}$$

Moreover,

$$|\widehat{\mathrm{d}S}_{uv} - \mathrm{d}S^\star_{uv}| \le \frac{G_O\rho(3V_{\max} + O_{\max})}{1-\rho}\, p^\star_{uv}. \tag{63}$$

Consequently, the omitted query-gradient contribution from the skipped block obeys

$$\big\|\Delta(\mathrm{d}q_u)^{(j)}\big\|_2 \le \frac{C_{\mathrm{score}}K_{\max}|J_j|\exp\big(-\Gamma^{\mathrm{fwd}}_{u,j}\big)}{1-\rho}, \tag{64}$$

and the omitted value-gradient contribution satisfies

$$\big\|\Delta(\mathrm{d}V_{J_j})\big\|_F \le \frac{G_O\sum_{u\in I_i}|J_j|\exp\big(-\Gamma^{\mathrm{fwd}}_{u,j}\big)}{1-\rho}. \tag{65}$$

If, in addition, $|A_u| \le A_{\max}$ and $|\Delta_v| \le \Delta_{\max}$ on the same block, then the omitted scalar-factor gradients are bounded by

$$\big|\Delta(\mathrm{d}A_u)^{(j)}\big| \le \frac{C_{\mathrm{score}}\Delta_{\max}|J_j|\exp\big(-\Gamma^{\mathrm{fwd}}_{u,j}\big)}{1-\rho}, \tag{66}$$

and

$$\big\|\Delta(\mathrm{d}\Delta_{J_j})\big\|_2 \le \frac{C_{\mathrm{score}}A_{\max}\sum_{u\in I_i}|J_j|\exp\big(-\Gamma^{\mathrm{fwd}}_{u,j}\big)}{1-\rho}. \tag{67}$$

Proof. By Equations equation 39, equation 48, and equation 51,

$$\widehat{p}_{uv} = \frac{\exp(s_{uv})}{\widehat{Z}_u} = \frac{\exp(s_{uv})}{Z_u(1-\delta_u)} = \frac{p^\star_{uv}}{1-\delta_u}. \tag{68}$$

Summing over $v \in J_j$ gives

$$\sum_{v\in J_j}\widehat{p}_{uv} = \frac{\varepsilon^{\mathrm{fwd}}_{u,j}}{1-\delta_u}, \tag{69}$$

and combining this with Equation equation 43 proves Equation equation 53. Under Equation equation 58, we also have $(1-\delta_u)^{-1} \le (1-\rho)^{-1}$.

For the score-gradient bound, the non-skip branch of Algorithm 2 uses

$$\widehat{\mathrm{d}S}_{uv} = \widehat{p}_{uv}\Big(\mathrm{d}o_u^\top v_v - \mathrm{d}o_u^\top \widehat{o}_u\Big). \tag{70}$$

Hence

$$|\widehat{\mathrm{d}S}_{uv}| \le \widehat{p}_{uv}\|\mathrm{d}o_u\|_2(\|v_v\|_2 + \|\widehat{o}_u\|_2) \le C_{\mathrm{score}}\widehat{p}_{uv}, \tag{71}$$

which proves Equation equation 62. For the gap to the dense score gradient,

$$\widehat{\mathrm{d}S}_{uv} - \mathrm{d}S^\star_{uv} = (\widehat{p}_{uv} - p^\star_{uv})\big(\mathrm{d}o_u^\top v_v - \mathrm{d}o_u^\top o^\star_u\big) - \widehat{p}_{uv}\,\mathrm{d}o_u^\top(\widehat{o}_u - o^\star_u). \tag{72}$$

Using Equation equation 68, the identity

$$\widehat{p}_{uv} - p^\star_{uv} = \frac{\delta_u}{1-\delta_u} p^\star_{uv}, \tag{73}$$

Equation equation 55, and the assumptions in Equations equation 58 and equation 59, we obtain

$$\begin{aligned} |\widehat{\mathrm{d}S}_{uv} - \mathrm{d}S^\star_{uv}| &\le \frac{\delta_u}{1-\delta_u} p^\star_{uv} G_O(\|v_v\|_2 + \|o^\star_u\|_2) + \widehat{p}_{uv} G_O \|\widehat{o}_u - o^\star_u\|_2 \\ &\le \frac{\delta_u}{1-\delta_u} p^\star_{uv} G_O(V_{\max} + O_{\max}) + \frac{p^\star_{uv}}{1-\delta_u} G_O(2V_{\max}\delta_u) \\ &\le \frac{G_O \rho (3V_{\max} + O_{\max})}{1-\rho}\, p^\star_{uv}, \end{aligned} \tag{74}$$

which proves Equation equation 63. Using the linear updates in Algorithm 2,

$$\Delta(\mathrm{d}q_u)^{(j)} = \sum_{v \in J_j} \widehat{\mathrm{d}S}_{uv} k_v, \qquad \Delta(\mathrm{d}V_{J_j}) = \widehat{P}^\top_{ij} \mathrm{d}O_i, \tag{75}$$

and similarly

$$\Delta(\mathrm{d}A_u)^{(j)} = \sum_{v \in J_j} \widehat{\mathrm{d}G}_{uv} \Delta_v, \qquad \Delta(\mathrm{d}\Delta_{J_j}) = \widehat{\mathrm{d}G}^\top_{ij} A_i, \tag{76}$$

with $|\widehat{\mathrm{d}G}_{uv}| \le |\widehat{\mathrm{d}S}_{uv}|$ because $\sigma(-A_u\Delta_v) \in (0,1)$. Applying the triangle inequality together with Equations equation 53, equation 62, and equation 58 yields Equations equation 64–equation 67. □

The backward bound above is best viewed as a hypothetical reconstruction result tied to the truncated log-partition actually stored by Algorithm 1. Algorithm 2 does *not* explicitly materialize $\widehat{p}_{uv}$ for skipped blocks; it simply skips them. What the analysis shows is that, if a forward-skipped block were reconstructed from the stored statistic $\widehat{\ell}_u$, the recovered probabilities would differ from the dense ones through the renormalization factor $(1-\delta_u)^{-1}$, while the reconstructed score gradients would additionally reflect the mismatch between the dense output $o^\star_u$ and the truncated output $\widehat{o}_u$. When the total skipped row mass is small, both effects remain controlled, which helps explain why continuing to skip such blocks in backward is often a reasonable approximation. At the same time, this argument does not by itself establish that the actual backward gate-based screening rule, with its own traversal order and running statistic, skips exactly those blocks or only those blocks.

D EXPERIMENT SETTINGS

D.1 GENERAL SETUP

To make the comparison between attention variants fair and reproducible, we standardize the model architecture, data pipeline, optimization, and evaluation protocol, changing only the attention module and its associated sparse-attention hyperparameters. All experiments are implemented with NVIDIA PyTorch container images (NVIDIA, 2022) and the Transformers framework (Wolf et al., 2020). Unless otherwise specified, we use the GPT-NeoX tokenizer (Black et al., 2022), the AdamW optimizer (Loshchilov and Hutter, 2017), and the WSD learning-rate schedule (Hägele et al., 2024), following compute-optimal scaling principles (Li et al., 2025; Hoffmann et al., 2022) on SmolLMCorpus (Ben Allal et al., 2024). For DMA pre-training, skipping is disabled for the first 2K optimization steps so that the gate predictor receives an initial period of stable dense-training supervision before sparse screening is activated. For evaluation, we use the EleutherAI LM Evaluation Harness (Gao et al., 2021) for perplexity measurements and LightEval (Fourrier et al., 2023) for downstream benchmarks.

Across module acceleration, MQAR, and scaling-law experiments, we compare five training-time attention variants: Full Attention (Vaswani et al., 2017), SWA (Fu et al., 2025), MLA (DeepSeek-AI et al., 2025), NSA (Yuan et al., 2025a), and DMA. For 1.7B evaluations, we additionally compare against inference-time sparse baselines including H2O (Zhang et al., 2023), InfLLM (Xiao et al., 2024), Quest (Tang et al., 2024a), DAM (Zhang et al., 2025), and Exact-Top. For continued-training experiments on Qwen3-8B, we compare DMA against full attention and sparse adaptation baselines including HySparse (Gao et al., 2026) and BLASST (Yuan et al., 2025b). Baseline-specific sparse hyperparameters are tuned to matched operating budgets using each method's original design principles. Under a fixed parameter budget, we keep the depth, width, and number of heads the same across variants and modify only attention-specific settings. For DMA, the reported w should be interpreted as an approximate effective retained-key budget induced by the threshold setting under the corresponding configuration, so that comparisons with window-based or budget-based sparse baselines reflect similar effective sparsity levels without introducing an additional top-w hyperparameter.

D.2 MODULE ACCELERATION

For ease of interpretation of Table 4, we compare the per-head theoretical complexities of different attention mechanisms under a unified assumption: default causal attention, fixed head dimension d_h, ignoring constant overheads such as QKV/output projection and normalization, and counting only attention-score computation together with the corresponding peak activation memory. This simplification is used only for the asymptotic summary in Table 4; the measured speed benchmarks discussed below are module-level end-to-end tests. Let n be the sequence length, w be the number of retained keys per head, d_c be the compression or latent dimension, B_c be NSA's compression block size, B_s be NSA's selection block size, and k be the number of selected blocks or entries. We use B_c and B_s here specifically to avoid overloading DMA's kernel tile size B from the main text. Under this convention, Full Attention is $O(n^2 d_h)$/$O(n^2)$; SWA is $O(nwd_h)$/$O(nw)$; MLA uses d_c instead of d_h under the same retained budget w, with complexity $O(nwd_c)$/$O(nw)$; NSA's composite terms come from the compression branch $O(n^2 d_c/B_c)$, selection branch $O(nkB_s d_h)$, and local branch $O(nwd_h)$, with memory $O(n^2/B_c + nkB_s)$; KV-selection methods (H2O/InfLLM/Quest/DAM) only compute on the selected k items, yielding $O(nkd_h)$/$O(nk)$; and DMA is of the same order as SWA, $O(nwd_h)$/$O(nw)$. We use this equivalent-budget view to estimate sparsity ratios and keep comparisons under similar effective computational budgets.

Table 5 lists the configurations used in the speed benchmark tests. We report representative settings, including the number of heads, head dimension, and sparse-attention hyperparameters for each method, to keep the comparison fair and reproducible. All experiments are conducted on an A100-SXM4-80GB GPU using bf16 precision, and each reported runtime is averaged over 100 runs after 25 warm-up iterations. Unlike the theoretical complexity summary in Table 4, the reported runtime here comes from module-level end-to-end measurements of the attention module and includes QKV projections together with any method-specific screening or heuristic overheads inside the module. For DMA, this overhead consists mainly of the lightweight A/Δ projections, per-block extrema updates,

Table 4: **Comparison of Different Attention Variants**. Comparison of different Self-Attention mechanisms. n denotes sequence length, d_h represents head dimension, w is window size, d_c is compressed dimension, B_c is NSA's compression block size, B_s is NSA's selection block size, and k is selection budget. Complexities focus on attention weight computation and memory requirements.

MECHANISM	COMPUTATION COMPLEXITY	MEMORY COMPLEXITY
Full Attention	$O(n^2 d_h)$	$O(n^2)$
SWA	$O(nwd_h)$	$O(nw)$
MLA	$O(nwd_c)$	$O(nw)$
NSA	$O(n^2 d_c/B_c + nkB_s d_h + nwd_h)$	$O(n^2/B_c + nkB_s)$
H2O	$O(nkd_h)$	$O(nk)$
InfLLM	$O(nkd_h)$	$O(nk)$
Quest	$O(nkd_h)$	$O(nk)$
DAM	$O(nkd_h)$	$O(nk)$
DMA	$O(nwd_h)$	$O(nw)$

Table 5: **Speed Benchmark Configurations**. The common settings for running time curves and the sparse hyperparameters for each method.

ALGO	warmups	runs	n_h	$n_{h_{kv}}$	d_h	w	d_c	B_c	B_s	k	precision
Full Attention[1]	25	100	32	8	128	-	-	-	-	-	bf16
SWA[2]	25	100	32	8	128	1024	-	-	-	-	bf16
MLA[3]	25	100	32	8	128	1024	192	-	-	-	bf16
NSA[4]	25	100	32	8	128	512	256	32	64	16	bf16
DMA	25	100	32	8	128	1024	-	-	-	-	bf16

and fused gate formation, all of which are small relative to the retained tile matmul-softmax-value path. We exclude only computations outside the attention module itself.

D.3 MULTI-QUERY ASSOCIATIVE RECALL

Following prior work (Arora et al., 2024), we construct a harder multi-query associative recall benchmark to isolate long-range retrieval under strong interference. The task uses longer sequences, smaller model dimensions, and 512 key-value pairs. All non-query/key/value positions are replaced with random tokens so that the model must recover the correct values through explicit lookup rather than contextual shortcuts. The dataset contains 250K training examples and 1K test examples, and all models are trained for 100 epochs. For a matched-budget comparison, sparse baselines use a fixed window size of 512, while DMA is tuned through its threshold to reach a comparable effective retained-key budget.

D.4 SCALING LAWS

We summarize the meaning of the columns in Table 6 and clarify which hyperparameters are used by each attention variant.

- **Params**: total number of model parameters.
- **Steps**: training steps.
- **Batch**: tokens per step.

[1]The implementation code for Full Attention is available at `github.com/Dao-AILab/flash-attention`.

[2]The implementation code for SWA is available at `github.com/Dao-AILab/flash-attention`.

[3]The implementation code for MLA is available at `github.com/deepseek-ai/FlashMLA`.

[4]The implementation code for NSA is available at `github.com/lucidrains/native-sparse-attention-pytorch`.

Table 6: **Self-Attention Variants Scaling Laws Configurations**. The model and hyperparameter configurations used in our self-attention variants scaling laws experiments.

ALGOS	PARAMS	STEPS	BATCH	LR	n_{layers}	d_{model}	n_h	$n_{h_{kv}}$	w	d_c	B_c	B_s	k
Full Attention	≈ 80M	13,500	0.128M tokens	3e-3	12	768	6	3	-	-	-	-	-
SWA	≈ 80M	13,500	0.128M tokens	3e-3	12	768	6	3	1024	-	-	-	-
MLA	≈ 80M	13,500	0.128M tokens	3e-3	12	768	6	3	1024	192	-	-	-
NSA	≈ 80M	13,500	0.128M tokens	3e-3	12	768	6	3	512	192	32	64	16
DMA	≈ 80M	13,500	0.128M tokens	3e-3	12	768	6	3	1024	-	-	-	-
Full Attention	≈ 200M	20,800	0.192M tokens	2e-3	16	1024	8	4	-	-	-	-	-
SWA	≈ 200M	20,800	0.192M tokens	2e-3	16	1024	8	4	1024	-	-	-	-
MLA	≈ 200M	20,800	0.192M tokens	2e-3	16	1024	8	4	1024	256	-	-	-
NSA	≈ 200M	20,800	0.192M tokens	2e-3	16	1024	8	4	512	192	32	64	16
DMA	≈ 200M	20,800	0.192M tokens	2e-3	16	1024	8	4	1024	-	-	-	-
Full Attention	≈ 680M	35,000	0.392M tokens	1e-3	24	1536	12	6	-	-	-	-	-
SWA	≈ 680M	35,000	0.392M tokens	1e-3	24	1536	12	6	1024	-	-	-	-
MLA	≈ 680M	35,000	0.392M tokens	1e-3	24	1536	12	6	1024	256	-	-	-
NSA	≈ 680M	35,000	0.392M tokens	1e-3	24	1536	12	6	512	192	32	64	16
DMA	≈ 680M	35,000	0.392M tokens	1e-3	24	1536	12	6	1024	-	-	-	-
Full Attention	≈ 1.7B	40,000	1M tokens	1e-3	28	2048	16	8	-	-	-	-	-
SWA	≈ 1.7B	40,000	1M tokens	1e-3	28	2048	16	8	2048	-	-	-	-
MLA	≈ 1.7B	40,000	1M tokens	1e-3	28	2048	16	8	2048	256	-	-	-
NSA	≈ 1.7B	40,000	1M tokens	1e-3	28	2048	16	8	512	256	32	64	16
DMA	≈ 1.7B	40,000	1M tokens	1e-3	28	2048	16	8	2048	-	-	-	-

- **LR**: peak learning rate.
- n_{layers}: number of Transformer layers.
- d_{model}: model hidden size.
- n_h: number of attention heads.
- $n_{h_{kv}}$: number of KV heads. In all our configurations we set $n_{h_{kv}} = n_h/2$.
- **Full Attention**: standard scaled dot-product attention. Sparse-specific columns (w, d_c, B_c, B_s, k) are not used.
- **SWA** (sliding-window attention): w is the sliding window size; other sparse-specific columns are not used.
- **MLA** (multi-head latent attention): d_c is the latent/compression dimension, w is the top-k budget; other columns are not used.
- **NSA** (native sparse attention): d_c is the compression dimension, w is the sliding window size, B_c is the compression block size, B_s is the selection block size, and k is the num selected blocks. All settings follow the original paper.
- **DMA** (dynamic mask attention): other sparse-specific columns are not used beyond the reported comparison budget.

D.5 BENCHMARK EVALUATION

We evaluate the largest 1.7B models selected from the scaling-law study under standard zero-shot and five-shot settings. The standard benchmark suite covers language modeling, commonsense reasoning, reading comprehension, and world knowledge through LAMBADA (Paperno et al., 2016), MMLU (Hendrycks et al., 2021), TriviaQA (Joshi et al., 2017), ARC (Clark et al., 2018), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), OBQA (Mihaylov et al., 2018), and WinoGrande (Sakaguchi et al., 2021). We then further fine-tune the same 1.7B models at 16K sequence length and increase the RoPE base frequency to 400K to strengthen long-context generalization. This longer-context evaluation combines standard reasoning and coding tasks with long-context benchmarks, covering MMLU (Hendrycks et al., 2021), BBH (Suzgun et al., 2023), GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2020), MBPP (Austin et al., 2021), LongBench (Bai et al., 2023), and RULER (Hsieh et al., 2024).

Evaluation randomness and aggregation. Table 1 uses deterministic likelihood scoring or greedy decoding for fixed checkpoints. For Table 2, we evaluate the fine-tuned checkpoints with decoding seeds 0, 1, and 2, using sampled generation (`do_sample=True`), temperature 0.6, and top-p 0.95 where generation is required. Deterministic task scoring is unchanged. We additionally evaluate SeerAttention (Gao et al., 2024) and FlexPrefill (Lai et al., 2025) in this benchmark suite. Each per-task entry gives the mean and standard deviation across the three decoding runs. The average column is the unweighted arithmetic mean of the seven task means; we do not infer an aggregate standard deviation from per-task deviations. These statistics quantify decoding sensitivity for each fixed checkpoint, rather than variation across independently trained models. Small mean differences are therefore interpreted as comparable performance rather than evidence of statistical superiority.

D.6 Length Extrapolation

For Needle-in-the-Haystack retrieval (Kamradt, 2023), we focus on Full Attention, NSA, and DMA under matched backbones and decoding settings. The benchmark inserts a short target span at varying depths inside long documents and measures whether the model can recover it as context length increases, including lengths beyond the nominal pre-training context window.

D.7 Sparse Adaptation via Continued Training

For sparse adaptation, we convert the dense Qwen3-8B backbone into sparse-attention models through continued training. During adaptation, we unfreeze only the attention modules and freeze the remaining model parameters. We continue training at 32K sequence length with a constant learning rate of 2×10^{-4} on 2B tokens, while keeping the remaining hyperparameters unchanged from the base model. The evaluation covers both long-input and long-output benchmarks: RULER (Hsieh et al., 2024) and LongBench (Bai et al., 2023) test long-context understanding, while MATH500 (Lightman et al., 2023), AIME 2024 (AI-MO, 2024), LiveCodeBench (Jain et al., 2024), and GPQA (Rein et al., 2024) probe reasoning, coding, and expert knowledge.

We also report end-to-end serving performance under a deployment-oriented setup with 8×A100 GPUs, data parallelism, 8 concurrent requests, and 8K input plus 24K output tokens, separating prefill throughput from decode throughput. For fair comparison, we follow each baseline's original hyperparameter recommendations, with the hybrid sparse baseline using a 1:3 ratio between full-attention and sparse-attention layers and the blocked sparse baseline targeting roughly 50% sparsity. The throughput measurements retain this 8K-input/24K-output serving workload; the long-context quality results in Table 3 use the separate evaluation protocol below.

D.8 Qwen3-8B Long-Context Subtask Results

After continued training at 32K for 2B tokens, we extend the evaluation context to approximately 131K tokens with YaRN, denoted 128K here. We evaluate RULER at this sequence length and LongBench with the same extended context configuration; individual LongBench inputs retain their task-specific lengths. For sampled generation, we use `do_sample=True`, temperature 0.6, top-p 0.95, and decoding seeds 0, 1, and 2. Tables 7 and 8 report every subtask's mean and standard deviation across these three decoding runs. These deviations measure fixed-checkpoint decoding sensitivity rather than independent training-run variability. Each macro mean is the unweighted average of the displayed subtask means; an aggregate standard deviation is not inferred from the subtask deviations. The six-metric Avg in Table 3 combines these two macro means with MATH500, AIME 2024, LiveCodeBench, and GPQA, excluding throughput.

This protocol measures length-extrapolation quality after 32K adaptation. The serving measurements retain the workload specified in Appendix D.7.

Table 7: **LongBench subtask results on Qwen3-8B.** The model uses the YaRN-extended 128K context configuration; individual examples need not fill the window. Entries are mean with standard deviation as a subscript over three decoding seeds. Bold and underlined means denote the highest and second-highest values without implying statistical significance.

Subtask	Full Attention	HySparse	BLASST	DMA
2WikiMQA	$\mathbf{9.3}_{\pm 0.83}$	$3.3_{\pm 0.54}$	$8.3_{\pm 0.84}$	$\underline{9.1}_{\pm 0.34}$
DuReader	$\mathbf{23.5}_{\pm 0.95}$	$21.2_{\pm 0.73}$	$21.7_{\pm 0.45}$	$\underline{23.2}_{\pm 0.93}$
GovReport	$\mathbf{31.1}_{\pm 0.34}$	$28.3_{\pm 0.34}$	$28.7_{\pm 0.86}$	$\underline{30.4}_{\pm 0.64}$
HotpotQA	$\mathbf{10.7}_{\pm 0.73}$	$3.5_{\pm 0.78}$	$7.7_{\pm 0.84}$	$\underline{10.5}_{\pm 0.87}$
LCC	$\mathbf{19.7}_{\pm 1.34}$	$16.3_{\pm 1.45}$	$17.1_{\pm 1.47}$	$\underline{18.2}_{\pm 1.24}$
LSHT	$44.7_{\pm 3.53}$	$42.2_{\pm 4.83}$	$\underline{44.9}_{\pm 3.78}$	$\mathbf{45.7}_{\pm 3.23}$
MultiNews	$\mathbf{24.9}_{\pm 0.43}$	$22.1_{\pm 0.94}$	$\underline{24.1}_{\pm 0.4}$	$23.8_{\pm 0.73}$
MFQA-en	$\underline{25.1}_{\pm 1.33}$	$23.9_{\pm 1.78}$	$24.9_{\pm 1.83}$	$\mathbf{25.4}_{\pm 1.62}$
MFQA-zh	$\mathbf{18.3}_{\pm 1.23}$	$13.1_{\pm 1.84}$	$16.3_{\pm 1.35}$	$\underline{17.7}_{\pm 1.73}$
MuSiQue	$6.6_{\pm 0.73}$	$\mathbf{9.3}_{\pm 0.89}$	$\underline{6.7}_{\pm 0.74}$	$5.7_{\pm 0.73}$
NarrativeQA	$\mathbf{3.8}_{\pm 0.23}$	$0.7_{\pm 0.2}$	$1.1_{\pm 0.34}$	$\underline{2.7}_{\pm 0.63}$
PsgCount	$\mathbf{3.3}_{\pm 1.13}$	$0.6_{\pm 0.94}$	$1.5_{\pm 0.34}$	$\underline{2.4}_{\pm 1.12}$
PsgRetr-en	$\underline{74.8}_{\pm 1.83}$	$70.4_{\pm 1.84}$	$73.5_{\pm 1.34}$	$\mathbf{75.3}_{\pm 1.13}$
PsgRetr-zh	$\mathbf{96.1}_{\pm 1.1}$	$85.4_{\pm 1.95}$	$93.3_{\pm 1.23}$	$\underline{94.7}_{\pm 1.52}$
Qasper	$\underline{11.9}_{\pm 0.82}$	$8.2_{\pm 0.85}$	$11.7_{\pm 0.84}$	$\mathbf{15.0}_{\pm 0.93}$
QMSum	$\mathbf{18.2}_{\pm 0.37}$	$13.6_{\pm 0.83}$	$16.2_{\pm 0.83}$	$\underline{17.7}_{\pm 0.72}$
RepoBench-P	$\mathbf{23.9}_{\pm 1.33}$	$20.1_{\pm 2.84}$	$22.8_{\pm 1.84}$	$\underline{22.9}_{\pm 1.62}$
Samsum	$\underline{44.3}_{\pm 1.28}$	$41.2_{\pm 1.35}$	$\underline{44.3}_{\pm 1.23}$	$\mathbf{45.1}_{\pm 1.13}$
TREC	$\mathbf{54.2}_{\pm 3.58}$	$49.2_{\pm 2.82}$	$52.2_{\pm 2.62}$	$\underline{52.7}_{\pm 2.37}$
TriviaQA	$\mathbf{91.1}_{\pm 2.12}$	$88.0_{\pm 1.84}$	$\underline{89.5}_{\pm 2.14}$	$\mathbf{91.1}_{\pm 1.72}$
VCSum	$\mathbf{13.8}_{\pm 0.3}$	$7.3_{\pm 0.73}$	$11.8_{\pm 0.23}$	$\underline{13.3}_{\pm 0.81}$
Macro mean	**30.92**	27.04	29.44	<u>30.60</u>

Table 8: **RULER-128K subtask results on Qwen3-8B.** All subtasks use the 128K length-extrapolation protocol. Entries are mean with standard deviation as a subscript over three decoding seeds. Bold and underlined means denote the highest and second-highest values without implying statistical significance.

Subtask	Full Attention	HySparse	BLASST	DMA
S1	$\mathbf{100.0}_{\pm 0.004}$	$98.6_{\pm 0.327}$	$99.2_{\pm 0.062}$	$\underline{99.8}_{\pm 0.012}$
S2	$72.6_{\pm 0.34}$	$57.0_{\pm 2.46}$	$\mathbf{95.6}_{\pm 2.91}$	$\underline{92.2}_{\pm 1.74}$
S3	$91.2_{\pm 0.29}$	$52.2_{\pm 0.83}$	$\underline{98.0}_{\pm 0.83}$	$\mathbf{99.2}_{\pm 0.83}$
MK1	$65.0_{\pm 1.52}$	$77.4_{\pm 3.34}$	$\underline{87.8}_{\pm 2.98}$	$\mathbf{89.2}_{\pm 2.82}$
MK2	$\underline{37.2}_{\pm 1.53}$	$12.8_{\pm 1.85}$	$\mathbf{71.8}_{\pm 1.92}$	$\mathbf{71.8}_{\pm 0.92}$
MK3	$0.8_{\pm 0.96}$	$0.004_{\pm 0.006}$	$\underline{7.2}_{\pm 3.92}$	$\mathbf{11.8}_{\pm 4.84}$
MQ	$77.0_{\pm 1.18}$	$70.1_{\pm 4.84}$	$\mathbf{88.7}_{\pm 3.23}$	$\underline{83.6}_{\pm 2.84}$
MV	$64.3_{\pm 1.73}$	$64.4_{\pm 2.88}$	$\underline{65.8}_{\pm 2.29}$	$\mathbf{66.6}_{\pm 2.18}$
VT	$91.6_{\pm 1.22}$	$90.4_{\pm 2.84}$	$\underline{92.4}_{\pm 2.57}$	$\mathbf{92.8}_{\pm 1.87}$
CWE	$4.8_{\pm 3.6}$	$0.002_{\pm 0.003}$	$\underline{5.8}_{\pm 1.34}$	$\mathbf{13.2}_{\pm 4.83}$
FWE	$77.6_{\pm 2.1}$	$68.7_{\pm 2.74}$	$\underline{80.6}_{\pm 2.72}$	$\mathbf{83.2}_{\pm 2.84}$
QA1	$\mathbf{42.5}_{\pm 0.528}$	$0.0715_{\pm 0.087}$	$34.7_{\pm 1.67}$	$\underline{39.8}_{\pm 0.834}$
QA2	$\mathbf{31.8}_{\pm 1.52}$	$0.56_{\pm 0.87}$	$29.4_{\pm 2.32}$	$\underline{31.2}_{\pm 1.98}$
Macro mean	58.18	45.56	<u>65.92</u>	**67.26**

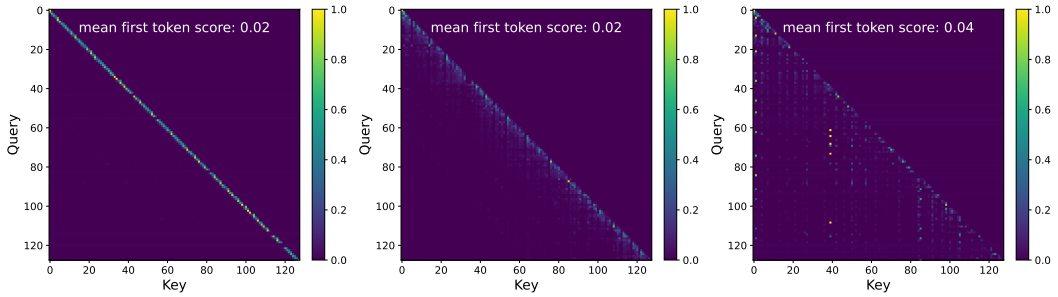

Figure 7: Representative attention patterns from different heads. These visualizations are taken from a model trained with DMA. Different heads exhibit distinct coarse-scale structures, including sharp diagonal locality, diffuse near-diagonal bands, and sparse off-diagonal activations. Although the token-level maps remain irregular and are not literally rank-1, their dominant mass follows relatively stable trends over $B \times B$ regions, which is exactly the granularity used by the attention kernel. This supports the use of lightweight factors $\mathbf{A}$ and $\mathbf{\Delta}$ as coarse block predictors: they do not need to reconstruct the full token-level map, only to anticipate which tiles are likely worth computing before those retained tiles are still evaluated exactly.

E Additional Empirical Analyses

Overview. The analyses below examine DMA's block-level screening design from complementary perspectives. Section E.1 visualizes attention patterns, and Section E.2 evaluates the threshold schedule. Section E.3 compares the learned rank-1 selector with an L2 score and higher-rank alternatives. Section E.4 measures forward–backward support agreement and numerical error during adaptation, while Section E.5 examines the effect of kernel tile geometry on latency and downstream quality.

E.1 Attention Pattern Visualization

Figure 7 visualizes representative attention maps from different heads of a model trained with DMA. These patterns are not intended to show that token-level attention is literally low-rank. Instead, they illustrate the weaker structure used by DMA: across heads, the dominant attention mass often forms coherent block-scale trends, such as local diagonal bands, broader near-diagonal regions, or sparse off-diagonal activations. This supports the modeling assumption in Section 2 and Appendix A: a lightweight rank-1 scalar gate can serve as a coarse predictor of which tiles are likely worth materializing, while the retained tiles are still evaluated with exact token-level attention.

E.2 Threshold Selection

Having established DMA's end-to-end performance, we isolate the impact of specific design choices from Section 2. We first examine the query-position-dependent margin in Equation 5. To make calibration comparable across context lengths, we parameterize the base margin as $\tau = c/N$ and report sweeps over the normalized coefficient c. Table 9 therefore evaluates schedules of the form $\theta_i = (c/N) \cdot (iB/N)$, where larger c yields more aggressive pruning. The results show a clear quality-efficiency trade-off: larger coefficients increase sparsity monotonically, but overly aggressive settings remove useful context and hurt RULER and GPQA. The dynamic schedule gives the best operating regime after calibration, retaining strong accuracy while skipping many blocks at long context, whereas large constant thresholds over-prune and very small constant thresholds behave nearly dense. In practice, c can be selected on a held-out set to meet a target sparsity or throughput budget and then reused across deployment lengths.

Computation allocation across layers and GQA positions. We further examine how DMA distributes attention computation on Qwen3-8B at 8K context length. We use RULER's MQ and MV subtasks for retrieval and QA1 and QA2 for question answering, with 500 examples per subtask and three decoding seeds. The model has 32 query heads and 8 KV heads, with four query heads per

Table 9: **Ablation on threshold selection.** This ablation tests how the threshold function controls the quality-efficiency trade-off. We report sparsity ratios from 4K to 32K context together with downstream performance on RULER and GPQA. Rows labeled by c use $\tau = c/N$ and hence the dynamic schedule $\theta_i = (c/N) \cdot (iB/N)$. The dynamic threshold preserves accuracy substantially better than overly aggressive constant thresholds while still delivering high sparsity at long context.

Coefficient	4K Context Sparsity ↑	8K Context Sparsity ↑	16K Context Sparsity ↑	32K Context Sparsity ↑	RULER acc ↑	GPQA acc ↑
$c = 128$	42.8%	65.5%	86.7%	98.2%	92.4	64.7
$c = 64$	37.1%	58.4%	74.2%	86.7%	92.7	65.4
$c = 32$	31.4%	51.2%	68.1%	81.7%	92.7	63.8
$c = 16$	25.1%	41.5%	58.4%	74.2%	92.9	63.5
$c = 8$	13.1%	25.1%	41.9%	58.4%	93.2	61.2
$\theta = 4 \cdot 10^{-3}$	86.2%	93.7%	96.7%	98.7%	56.2	59.3
$\theta = 4 \cdot 10^{-5}$	2.9%	4.7%	6.9%	8.1%	93.2	61.2
Dense	0%	0%	0%	0%	93.2	61.2

Table 10: **Attention sparsity across layers and GQA positions.** Entries are mean sparsity percentages on retrieval (MQ/MV) and question-answering (QA1/QA2) tasks at $N = 8192$. Higher sparsity means less retained attention computation. G_r averages the query heads at relative position r across the eight KV groups. Panel headers retain the threshold labels used in this sweep and report full-model RULER-8K mean ± standard deviation over three decoding seeds.

	Retrieval sparsity (%)				QA sparsity (%)			
Layer	G_0	G_1	G_2	G_3	G_0	G_1	G_2	G_3
Threshold setting: $1/N$; RULER-8K: 92.52 ± 0.63								
0	60.40	63.87	60.39	57.52	73.81	71.69	80.52	68.18
11	67.60	62.44	65.41	62.72	76.88	71.50	83.71	70.06
23	74.47	80.41	76.93	77.37	84.05	82.15	91.99	79.08
35	68.05	61.90	65.50	72.73	75.77	69.73	83.09	76.59
Threshold setting: $2/N$; RULER-8K: 91.88 ± 0.83								
0	63.02	66.96	62.53	59.66	77.07	74.77	85.60	71.81
11	70.04	65.82	68.05	66.66	80.57	74.27	86.85	72.50
23	78.18	83.50	80.62	80.57	88.75	87.28	92.17	83.68
35	71.90	65.34	68.25	77.26	79.30	72.56	87.39	80.71
Threshold setting: $4/N$; RULER-8K: 91.42 ± 0.78								
0	65.92	70.10	66.60	62.71	81.46	79.74	88.16	74.84
11	73.69	69.18	72.56	69.75	84.76	79.30	91.75	77.58
23	82.38	87.52	84.13	85.61	91.88	91.05	92.54	86.94
35	74.40	67.51	71.30	80.30	84.22	77.24	92.16	84.82
Threshold setting: $8/N$; RULER-8K: 91.18 ± 0.64								
0	67.32	70.92	67.97	65.62	82.15	80.48	89.72	77.72
11	77.13	69.32	73.94	71.52	87.11	81.40	92.87	78.33
23	84.75	90.38	86.70	87.03	93.23	92.37	93.50	88.33
35	76.84	70.65	74.18	81.61	85.53	78.40	94.62	86.61

KV group. We define G_0–G_3 by a query head's relative position within its KV group and average each position across the eight groups. Thus, these columns summarize relative GQA positions rather than individual heads. Table 10 reports mean sparsity for layers 0, 11, 23, and 35 at four threshold settings, together with the corresponding full-model RULER-8K scores. The shared gate projection parameters are applied to each head's own query/key representations, allowing the resulting routing decisions to vary across heads.

Retrieval uses denser attention than QA at every matched layer and GQA position in this sweep. Allocation also varies with depth: layer 23 is generally the sparsest of the four sampled layers, while sparsity does not increase monotonically from the first to the last layer. Differences across G_0–G_3

Table 11: **Ablation on selector rank and L2 scoring at 8K.** Sparse selectors operate at similar realized sparsity. Forward/backward latency and peak memory are kernel-level measurements; RULER is evaluated on the full model with three decoding seeds. Output error is measured against the corresponding screening-disabled operator.

Selector	Sparsity (%)	Recall@K (%)	Max. output error	Fwd/Bwd (ms)	Fwd/Bwd memory (MiB)	RULER-8K
Full Attention	–	–	–	2.83 / 8.27	161 / 515	92.38 ± 0.32
L2	62.1	49.46	5×10^{-2}	1.17 / 3.17	162 / 516	74.61 ± 0.72
Rank-1	62.3	99.13	2×10^{-4}	1.15 / 3.08	162 / 518	92.52 ± 0.63
Rank-2	62.2	99.35	1×10^{-4}	1.52 / 3.92	162 / 520	92.13 ± 0.34
Rank-4	62.3	99.63	3×10^{-4}	1.58 / 4.08	164 / 525	92.65 ± 0.70
Rank-8	62.2	99.62	4×10^{-4}	1.64 / 4.23	166 / 535	92.52 ± 0.85
Rank-16	62.5	99.84	7×10^{-5}	1.67 / 4.35	171 / 555	92.82 ± 0.87

show that shared projection parameters can produce head-dependent routing. Increasing the threshold setting raises sparsity throughout these groups, while mean RULER-8K decreases from 92.52 to 91.18. These observations characterize the sampled layers and averaged GQA positions; they do not establish distinct functions for individual heads.

E.3 Selector Rank and L2 Ablation

We compare DMA's learned rank-1 selector with an L2-magnitude selector and learned rank-r alternatives for $r \in \{2, 4, 8, 16\}$ on Qwen3-8B. The learned selectors are evaluated after continued training for 2B tokens at 8K context length. The L2 selector provides a simple magnitude-based scoring baseline without learned selector parameters. This ablation uses an 8K adaptation setting, distinct from the 32K sparse-adaptation experiments in Section D.7.

Measurement protocol. The kernel measurements in this section and Sections E.4–E.5 use 6,500 first-layer $Q/K/V$ states exported from RULER evaluations, with sequence lengths $N \in \{1024, 2048, 4096, 8192\}$. We use an NVIDIA A100-SXM4-80GB GPU, batch size 1, 32 query heads, 8 KV heads, and head dimension 128. Latencies are averaged after 20 warm-up iterations. These measurements evaluate kernels on exported states, rather than the full attention module including QKV projections. RULER-8K scores are full-model results reported as mean $\pm$ standard deviation over decoding seeds 0, 1, and 2, with temperature 0.6 and top-p 0.95; these deviations measure decoding sensitivity for fixed checkpoints, not training-run variance.

Selection fidelity. For each query/head, let S_{sel} be the retained block set and let S_{QK} be the full-QK Exact-Top-K block set, where $K = |S_{\text{sel}}|$. We report

$$\text{Recall@}\, K = \frac{|S_{\text{sel}} \cap S_{QK}|}{K}. \tag{77}$$

Output and gradient errors compare sparse execution with the same checkpoint and attention operator with screening disabled, retaining all causal tiles. Thus the reference preserves the gate and isolates the effect of sparse execution. The tables retain the maximum-error metrics reported by this evaluation protocol; they do not establish worst-case bounds across layers or inputs.

Table 11 shows that rank 1 recovers 99.13% of the full-QK top-ranked blocks at 62.3% sparsity, compared with 49.46% recall for L2 at 62.1% sparsity. RULER improves from 74.61 for L2 to 92.52 for rank 1, close to Full Attention's 92.38. Increasing the rank to 16 adds 0.71 percentage points of recall and 0.30 RULER points, while increasing forward and backward latency by approximately 45% and 41%, respectively. These results support rank 1 as a lower-cost operating point among the tested learned selectors, without establishing a significant downstream advantage over higher ranks.

Across the four measured lengths, rank-1 recall remains above 99.1% (Table 12). This provides empirical evidence for the selector at these operating points. It does not turn the gate-only upper bound into a bound on unseen QK scores, nor does it establish that the dominant-direction assumption holds for every head.

Table 12: **Rank-1 selection fidelity across sequence lengths.** Measurements use the final adaptation checkpoint and the same first-layer protocol as Table 11.

Length	Sparsity (%)	Recall@K (%)	Max. output error	Fwd/Bwd (ms)
1K	29.1	99.13	1×10^{-3}	0.09 / 0.41
2K	39.7	99.23	7×10^{-4}	0.17 / 0.60
4K	43.3	99.32	5×10^{-4}	0.39 / 1.51
8K	62.3	99.13	2×10^{-4}	1.15 / 3.08

Table 13: **Forward–backward consistency at three adaptation stages.** Stage denotes the number of adaptation tokens processed. Errors compare sparse execution with the corresponding dense gated operator, rather than with an exact backward pass on the forward-retained support.

Length	Stage	IoU (%)	Max. $\mathbf{dQ}$ error	Max. $\mathbf{dK}$ error	Max. $\mathbf{dV}$ error
1K	200M	98.82	3×10^{-3}	1×10^{-3}	4×10^{-4}
1K	1B	99.82	2×10^{-2}	8×10^{-4}	5×10^{-4}
1K	2B	99.76	2×10^{-3}	9×10^{-4}	4×10^{-4}
2K	200M	97.91	1×10^{-3}	3×10^{-4}	2×10^{-4}
2K	1B	99.72	8×10^{-4}	2×10^{-4}	1×10^{-4}
2K	2B	99.21	9×10^{-4}	2×10^{-4}	1×10^{-4}
4K	200M	98.62	8×10^{-4}	1×10^{-4}	6×10^{-5}
4K	1B	98.71	8×10^{-4}	2×10^{-4}	6×10^{-5}
4K	2B	99.33	5×10^{-4}	1×10^{-4}	6×10^{-5}
8K	200M	99.16	7×10^{-4}	9×10^{-5}	2×10^{-5}
8K	1B	99.63	2×10^{-4}	7×10^{-5}	2×10^{-5}
8K	2B	99.76	4×10^{-4}	7×10^{-5}	2×10^{-5}

E.4 Forward–Backward Consistency During Adaptation

We evaluate the traversal-dependent backward approximation described in Appendix B. Forward screening maintains a query-side reference, whereas the key-major backward procedure maintains a key-side reference; the two procedures can therefore select different tiles. For their retained supports M_f and M_b, we measure

$$\mathrm{IoU}(M_f, M_b) = \frac{|M_f \cap M_b|}{|M_f \cup M_b|}. \tag{78}$$

We sample Qwen3-8B checkpoints after 200M, 1B, and 2B adaptation tokens. The gate projection vectors are initialized to all ones, so their initial projections sum Q/K features along the head dimension. Table 13 reports support agreement and the maximum gradient-error metrics relative to the screening-disabled operator under the first-layer protocol above.

IoU ranges from 97.91% to 99.82% across the sampled settings. The largest reported $\mathbf{dQ}$ error is 2×10^{-2} at 1K in the middle stage; the largest $\mathbf{dK}$ and $\mathbf{dV}$ errors in the table are 1×10^{-3} and 5×10^{-4}. At 8K, the final checkpoint has 99.76% support IoU. These observations quantify high support overlap under the tested conditions, but neither imply identical masks nor prove exact gradients or convergence. In particular, the first sampled checkpoint is already 200M tokens into adaptation, and these measurements do not cover all layers or the initialization step.

E.5 Kernel Tile Geometry Ablation

We vary the forward and backward tile geometries on the target A100 GPU. Here (B_q, B_k) denotes the query and key dimensions of a tile; rectangular tiles generalize the common block size B used in the pseudocode. The selected forward configuration is $(128, 32)$ with 4 warps, and the selected backward configuration is $(32, 128)$ with 8 warps. For the downstream comparison, we keep the checkpoint and generation settings fixed and change only the inference forward tile. Backward tile

Table 14: **Ablation on kernel tile geometry at 8K.** Latencies and recall follow the first-layer kernel protocol. RULER-8K reports full-model mean $\pm$ standard deviation over three decoding seeds. The reported maximum output-error metric is 2×10^{-4} for all four geometries.

Forward tile	Backward tile	Fwd (ms)	Bwd (ms)	Recall@K (%)	RULER-8K
128×32	32×128	1.15	3.08	99.13	92.52 ± 0.63
64×64	64×64	1.40	3.95	99.27	92.39 ± 0.61
64×32	32×64	1.71	4.69	99.57	92.58 ± 0.57
32×32	32×32	2.18	5.93	99.75	92.57 ± 0.60

choices affect the measured backward latency; the comparison does not retrain a model for each geometry.

Table 14 shows a 0.19-point range in mean RULER score across the tested geometries, compared with decoding-seed standard deviations of 0.57–0.63. The finest 32×32 geometry increases recall by 0.62 percentage points but raises forward and backward latency to approximately $1.90\times$ and $1.93\times$ the selected configuration. The corresponding maximum $\mathbf{dQ}$ error metric increases from 4×10^{-4} to 9×10^{-4}, while the output-error metric remains unchanged. These results support the selected tile geometry as an efficient operating point for this GPU and checkpoint; they do not establish invariance to tile size during training or on other hardware.