MASSALLOC ATTENTION: LET ATTENTION ALLOCATE ITS OWN COMPUTE

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-context full softmax attention (FullAttn) often assigns negligible normalized mass to much of the causal score space, yet dense kernels execute the complete post-score path after forming each QK tile. We introduce **MassAlloc Attention (MALA)**, a fused attention primitive that preserves score access to every legal causal interaction and uses normalized contribution to allocate post-score computation. Forward uses its evolving online-softmax normalizer, while backward reuses the finalized normalizer to derive nested retained support using only standard attention state. A common tolerance governs training and inference, allowing the retained work to adapt across queries, heads, layers, and inputs. MALA reduces low-contribution post-score computation while retaining quadratic QK score discovery. A matched-work study at 8K isolates the benefit of distribution-adaptive allocation: under exactly matched total post-score work, MALA approaches a per-instance reference-mass oracle, with mean omitted mass of 0.0188% versus 0.0182%, while static allocations perform substantially worse. Across context lengths from 1K to 32K tokens, the same tolerance maintains low output and gradient errors relative to the reference operator. Across a broader controlled associative-recall comparison, MALA closely tracks FullAttn as context grows, reaching 89.67% accuracy at 8K compared with 89.97% for FullAttn. In an attention-operator benchmark at 128K tokens on 8 H100 GPUs with tensor parallelism, MALA reduces forward and backward latency during training by $2.2\times$ and $3.0\times$ and decoding latency during inference by $1.6\times$ relative to FullAttn, while retaining FullAttn-level per-rank peak operator memory. Across scaling-law training from 0.6B to 14B parameters on 128 H100 GPUs, MALA closely tracks FullAttn in perplexity while reducing total training FLOPs, with a 23.1% reduction at 14B during 32K-context training. The resulting 14B models and 32B models from separate continued training achieve comparable knowledge, reasoning, and long-context retrieval scores to FullAttn. These results indicate that allocating post-score computation according to normalized attention contributions can retain the evaluated capabilities of FullAttn while reducing attention computation.

1 INTRODUCTION

Context lengths are expanding from thousands to hundreds of thousands of tokens (Snell et al., 2024) to support long-document understanding (Park et al., 2023; DeepMind, 2025), multi-turn reasoning (HuggingFace, 2025; Guo et al., 2025; Team, 2025), and repository-level code generation (Zhang et al., 2024). Attention over these long contexts incurs substantial computation and memory traffic. FlashAttention (Dao et al., 2022; Shah et al., 2024) improves the IO efficiency of self-attention (Vaswani et al., 2017) through tiling, fusion, and online softmax (Milakov & Gimelshein, 2018). However, these IO improvements leave the dense execution pattern unchanged: every legal causal tile still incurs post-score computation after its $\mathbf{QK}$ scores are formed, including softmax updates, $\mathbf{V}$ loading, and $\mathbf{PV}$ accumulation in forward, and probability reconstruction and the computation of $\mathbf{dP}$, $\mathbf{dS}$, $\mathbf{dQ}$, $\mathbf{dK}$, and $\mathbf{dV}$ in backward.

Full softmax attention (FullAttn) is highly non-uniform: its mass often concentrates in local regions, sink tokens, and a sparse set of retrieval-relevant interactions (Gu et al., 2024; Barbero et al., 2025; de Llano et al., 2025; Xiao et al., 2024b). Many remaining interactions receive negligible normalized mass (Gao et al., 2024; Yuan et al., 2025a), yet dense kernels still execute the full forward and

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.

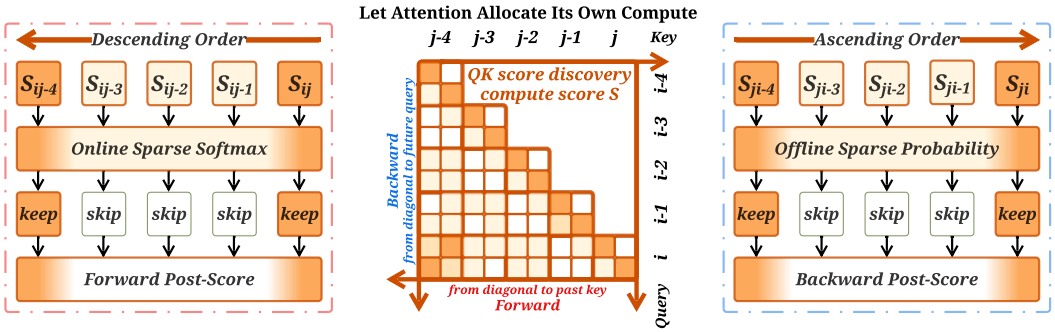

Figure 1: **MALA lets attention allocate its own post-score compute.** Every legal causal tile undergoes QK score discovery. Forward uses its evolving softmax normalizer to allocate forward post-score execution; backward reuses the finalized normalizer to derive nested retained support while storing only standard attention state.

backward post-score computation for low-contribution tiles. Attention already produces scores and softmax statistics that can guide these decisions. These observations motivate preserving score access to the complete causal context while using normalized contribution to decide which tiles warrant further computation, allowing attention to allocate its own post-score compute.

Allocating this work also requires an execution structure that remains efficient during training and inference. Static windows and block patterns prescribe a position-defined support (Child et al., 2019; Beltagy et al., 2020; Zaheer et al., 2020; Fu et al., 2025), while dynamic methods select tokens or blocks through content-dependent scores or routing (Tang et al., 2024; Lai et al., 2025; Li et al., 2024; Zhang et al., 2023; Xiao et al., 2024a; Qi et al., 2026; Zhao et al., 2025; Yuan et al., 2025b; Lu et al., 2025; Gao et al., 2024). Adaptive top-p methods vary the selected support with an estimated mass target (Lin et al., 2025; Ni et al., 2026). These approaches can avoid QK computation for excluded interactions; a complementary opportunity is to allocate post-score work inside the attention loop after forming each exact QK tile. This raises the question: can allocating post-score computation from attention's own normalized contributions preserve operator fidelity, model quality, and long-range retrieval while reducing the cost of training and inference?

We introduce **MassAlloc Attention (MALA)**, a fused attention primitive that converts normalized contribution into runtime compute allocation, as illustrated in Figure 1. Every legal causal tile first undergoes QK score discovery. During forward, MALA bounds normalized contributions using its evolving online-softmax normalizer; during backward, it uses the finalized normalizer saved by forward to evaluate each recomputed tile. In either pass, tiles whose contributions fall below a length-normalized tolerance omit their post-score work. The tolerance controls admissible contribution, while the realized attention distribution determines the retained computation across queries, heads, layers, and inputs.

The same allocation rule also supports fused execution for training and inference. Forward bounds the total dense probability mass assigned to omitted positions, while backward derives retained support nested within the forward support using only the saved normalizer. The operators use standard attention state without materializing a selection mask or retained-tile indices. A common tolerance governs the training forward and backward passes, inference prefill, and autoregressive decoding, while the realized work remains specific to each execution context.

MALA retains quadratic QK score discovery; its savings arise from avoiding low-contribution post-score execution. MALA develops this execution direction into a normalized-mass allocation rule with paired forward and backward semantics. The forward rule provides an omitted-mass guarantee, while output and gradient fidelity are evaluated empirically.

We evaluate MALA around two questions: whether normalized-mass allocation preserves operator fidelity, language-model quality, and long-range retrieval, and whether the resulting reduction in post-score work translates into practical training and inference efficiency under tensor parallelism (Shoeybi et al., 2019). A matched-work allocation study first isolates the benefit of distribution-adaptive allocation: under exactly matched total post-score work, MALA approaches the per-instance reference-mass oracle, with mean omitted mass of 0.0188% versus 0.0182%, while static position- and layer-head-based allocations perform substantially worse. With the same tolerance from 1K to 32K tokens, MALA maintains low relative output and gradient errors against the reference operator. Across

a broader controlled associative-recall comparison, fixed-budget sparse baselines use a 1,024-slot ceiling while MALA adaptively allocates post-score work at a similar scale; at 8K, it reaches 89.67% accuracy, compared with 89.97% for FullAttn. In the attention-operator benchmark at 128K sequence length on 8 H100 GPUs with tensor parallelism, MALA reduces forward and backward latency during training by $2.2\times$ and $3.0\times$, and decoding latency during inference by $1.6\times$ relative to FullAttn, while retaining FullAttn-level peak operator memory. Across scaling-law training from 0.6B to 14B parameters, MALA closely tracks FullAttn in perplexity; at 14B, it reduces total training FLOPs by 2.5% during 4K pre-training and 23.1% during 32K long-context training. At the model level, the resulting 14B models and separately continued-trained 32B models retain aggregate knowledge, reasoning, and long-context retrieval performance comparable to FullAttn. Our contributions are as follows:

- We formulate sparse attention as **distribution-conditioned compute allocation**: every causal interaction remains score-accessible, while the probability distribution realized by attention determines its post-score work.
- We implement **MassAlloc Attention** for training and inference with paired online forward and offline backward rules and a common normalized-mass tolerance. The operators use only native attention state, with a forward omitted-mass bound and nested backward support.
- We evaluate the modeling and efficiency consequences of normalized-mass allocation through matched-work controls, operator fidelity, controlled associative recall, kernel execution, scaling-law training, and model-level evaluation.

2 METHODOLOGY

We formulate attention as a runtime compute allocator by separating QK score discovery from post-score execution. We first define the allocation rule and its contribution tolerance, then present the online forward and offline backward tests, characterize their computational cost, and describe fused execution for training and inference.

2.1 ATTENTION AS A COMPUTE ALLOCATOR

MALA computes QK scores for every legal causal tile and decides whether to execute the subsequent forward or backward computation from those scores and the available softmax state.

For notational simplicity, we describe a single query head and its associated key/value head, suppressing batch and head indices. Let N_q and N_k be the query and key sequence lengths, with $\mathbf{Q} \in \mathbb{R}^{N_q \times d_h}$ and $\mathbf{K}, \mathbf{V} \in \mathbb{R}^{N_k \times d_h}$. We absorb the standard softmax scale $1/\sqrt{d_h}$ into $\mathbf{Q}$ and use natural exponentials and logarithms throughout. Queries are divided into blocks I_i of B_q rows and keys and values into blocks J_j of B_k rows. Lowercase q and k index individual query and key positions, while i and j index blocks.

For each tile, score discovery forms $\mathbf{S}_{ij} = \mathbf{Q}_i \mathbf{K}_j^\top$. Forward post-score computation comprises the softmax update, $\mathbf{V}$ loading, and $\mathbf{PV}$ accumulation; backward post-score computation comprises probability reconstruction and the computation of $\mathbf{dP}$, $\mathbf{dS}$, $\mathbf{dQ}$, $\mathbf{dK}$, and $\mathbf{dV}$. Here, $\mathbf{P}$ denotes attention probabilities, and $\mathbf{dX}$ denotes the backpropagated quantity associated with $\mathbf{X}$. FullAttn executes both stages for every legal tile, whereas MALA allocates post-score work after score discovery.

To define a common contribution scale, let $\mathcal{C}_q$ be the set of causally visible keys for query row q and let $L_q = |\mathcal{C}_q|$. Uniform attention assigns probability $1/L_q$ to each such key. We scale this reference by a shared dimensionless tolerance $\tau > 0$, giving the length-aware threshold τ/L_q. MALA compares the largest per-key contribution ratio in each tile with this threshold.

With zero-based token positions, $L_q = q + 1$ in equal-length causal training. In single-token autoregressive decoding, $L_0 = N_k$, including the current token. For block I_i, $\mathbf{L}_i \in \mathbb{R}^{B_q}$ collects these per-row counts over valid query rows. The normalization by L_q gives the tolerance the same interpretation across query positions and sequence lengths without separately calibrated thresholds. The tolerance specifies admissible contribution rather than a compute budget: sharp attention rows can reject many tiles, whereas diffuse rows generally retain more.

2.2 Online and Offline Allocation

Online Sparse Softmax. The forward pass (Algorithm 1) maintains the standard online-softmax row maximum $\mathbf{m}_i$ and shifted normalization sum ℓ_i over retained tiles. For query row q, let $R_q^{(t)}$ contain the causal keys retained before the current tile is tested, and let s_{qk} be the QK score for key k. The retained normalizer at this point is

$$Z_q^{(t)} := \sum_{k \in R_q^{(t)}} \exp(s_{qk}) = \exp\big(m_q^{(t)}\big)\ell_q^{(t)} \tag{1}$$

Here, $m_q^{(t)}$ and $\ell_q^{(t)}$ are the entries of the running block states for row q. For a nonempty retained set, $m_q^{(t)} + \log \ell_q^{(t)} = \log Z_q^{(t)}$. After forming a candidate tile, its largest contribution ratio for this row is

$$r_{qj}^{(t)} := \frac{\exp\big(\max_{k \in J_j \cap \mathcal{C}_q} s_{qk}\big)}{Z_q^{(t)}} \tag{2}$$

The denominator includes only the mass retained so far, so this ratio can exceed one and upper-bounds the largest dense-softmax probability in the candidate tile for that row.

The tile is skipped only when $r_{qj}^{(t)} < \tau / L_q$ for every valid query row with a legal entry in the tile. Taking logarithms and collecting the rows of I_i gives the blockwise test used by the kernel:

$$\mathrm{rowmax}(\mathbf{S}_{ij}) - \mathbf{m}_i - \log \ell_i < \log(\tau / \mathbf{L}_i) \tag{3}$$

The inequality is interpreted elementwise over relevant rows. Causally masked entries have score $-\infty$, and rows with no legal key in the candidate tile do not constrain its retention. The test is conservative at tile granularity: one row that fails the skip condition retains the tile for all rows in the query block.

Before any key has been retained for a row, $\ell_q^{(t)} = 0$ and $Z_q^{(t)} = 0$. For a candidate tile containing a legal key for that row, the log-ratio is defined as $+\infty$, so its first legal tile is necessarily retained and every forward attention row is nonempty. For causal attention, MALA visits tiles from the diagonal toward earlier keys. This traversal initializes the running state from valid recent context and typically establishes a strong normalizer early. Locality is an execution prior rather than a support assumption: all earlier causal tiles still undergo score discovery, and any distant tile with a sufficiently large score is retained.

A skipped tile leaves $\mathbf{m}_i$, ℓ_i, and $\mathbf{O}_i$ unchanged; a retained tile follows the standard online-softmax update. The final output is therefore ordinary softmax attention renormalized over the retained tiles. Let R_q denote the final retained set and $Z_{R,q} = \sum_{k \in R_q} \exp(s_{qk})$ its unnormalized mass. At the end of forward, the kernel overwrites the running sum buffer with the final log-normalizer, $\ell_i \leftarrow \mathbf{m}_i + \log \ell_i$, whose entries are $z_q = \log Z_{R,q}$. This saved state is passed to backward. Appendix B provides the forward approximation analysis.

Offline Sparse Probability. The backward kernel (Algorithm 2) uses the finalized normalizer instead of replaying the forward sequence of partial normalizers. Here, *offline* means that the normalizer is already finalized when a tile is tested; allocation still occurs at runtime and requires no offline calibration.

Backward applies the same contribution tolerance using $Z_{R,q}$ in place of the evolving $Z_q^{(t)}$. In the key-major traversal, the recomputed score tile is $\mathbf{S}_{ji} = \mathbf{K}_j \mathbf{Q}_i^\top = \mathbf{S}_{ij}^\top \in \mathbb{R}^{B_k \times B_q}$. Each query's saved log-normalizer and threshold are therefore broadcast along the key dimension. The resulting skip condition is

$$\mathbf{S}_{ji} - \ell_i^\top < \log(\tau / \mathbf{L}_i)^\top \tag{4}$$

Here the inequality is interpreted elementwise over valid causal entries and must hold throughout the tile. Otherwise, MALA reconstructs $\mathbf{P}_{ji} = \exp(\mathbf{S}_{ji} - \ell_i^\top)$ and executes the standard $\mathbf{dV}$, $\mathbf{dP}$, softmax-backward, $\mathbf{dQ}$, and $\mathbf{dK}$ path. Unlike forward, the backward rule does not require the diagonal tile to be retained.

In exact arithmetic, $Z_{R,q} \geq Z_q^{(t)}$ for every online test with positive retained mass, so using the finalized denominator cannot increase any key's contribution ratio. With the same tile partition and

causal mask in both passes, every forward-skipped tile therefore also satisfies the backward skip condition. Backward may additionally omit a tile that forward retained before its normalizer was complete. Its retained support is thus nested within the forward support without a stored forward mask.

This nesting does not imply exact differentiation of the forward operator, since backward can omit additional gradient contributions. Their error also depends on $\mathbf{dO}$ and $\mathbf{V}$, so we evaluate gradient fidelity directly at the operator level.

2.3 Attention Cost

For fixed head configuration, MALA remains quadratic in sequence length because it computes QK scores for every legal causal tile. Its savings are data-dependent constant-factor reductions in post-score arithmetic and memory traffic: a forward skip saves full-tile exponentiation, $\mathbf{V}$ loading, and $\mathbf{PV}$ accumulation; a backward skip saves probability reconstruction and the matrix multiplications and elementwise operations used to form $\mathbf{dV}$, $\mathbf{dP}$, $\mathbf{dS}$, $\mathbf{dQ}$, and $\mathbf{dK}$. The tests require a maximum reduction and scalar comparisons using existing attention state. Support-sparse methods can additionally avoid QK computation for excluded regions, whereas MALA retains full score accessibility. As $\tau \to 0$, the skip conditions vanish and both passes recover FullAttn.

We measure allocated work as the number of key slots per query for which the post-score path is executed. Retaining a $B_q \times B_k$ tile contributes B_k post-score key slots to each of its B_q query rows. We report the average as *mean post-score key slots per query*, excluding QK score discovery. This quantity captures realized work across training and inference, including the effect of tile sharing across rows. When attention architectures use different value dimensions, we separately use post-score FLOP-equivalent work for cross-architecture matching; Appendix C.1 defines this accounting.

2.4 Fused Execution

The allocation tests are embedded in the ordinary tiled attention loop. The running maximum, shifted normalization sum, and output accumulator remain on chip during forward. Forward writes only the output $\mathbf{O}$ and the final log-normalizer ℓ, while backward reuses ℓ to reconstruct retained probabilities. The fused operators materialize neither the attention matrix nor a binary mask, retained-tile indices, or router state. The same forward operator serves training and inference prefill; Appendix A.1 details the traversal and state handling for each pass.

The same probability tolerance governs training forward and backward, inference prefill, and decoding. Under split-KV decoding, each split tests tiles against its own partial normalizer, while L_q in τ/L_q still counts the query's full causal context. A smaller available normalizer makes the test more conservative and may retain additional post-score work, so a shared tolerance does not require split and unsplit execution to retain identical supports. Partial outputs are combined through the standard normalizer-aware reduction. MALA neither compresses nor evicts the persistent KV cache, so its decoding-memory scope is the attention operator's working set rather than cache capacity.

3 Experiments

Evaluation Overview. We organize the experiments around a sequence of increasingly comprehensive questions: whether the realized attention distribution improves compute allocation under fixed post-score work, whether one tolerance preserves the reference operator across context lengths, whether the resulting allocation supports arbitrary long-range associations and efficient operator execution, and whether its quality-compute trade-off persists across model scales. We conclude with knowledge, reasoning, and long-context evaluations of the resulting 14B models and of 32B models trained in a separate continued-training study.

Experimental Settings. Unless stated otherwise, MALA uses the same canonical tolerance, $\tau = 1$, across forward, backward, prefill, and decoding; the realized distribution sets the retained work for each layer, head, input, and sequence length. Within each experiment, attention variants use matched model scale, depth, hidden size, data, and optimization settings while retaining their native attention configurations. The matched-work and operator-fidelity studies use the same 14B MALA checkpoint obtained after the 32K long-context stage of the scaling-law study. Scaling-law and 32B continued

Table 1: **Matched-work allocation.** Every policy executes exactly the same total post-score work, with a shared average of approximately 1,024 slots per query. The first three rows use finalized per-instance reference mass for region selection and differ only in how work is allocated across decisions. Omitted probability mass and relative output L_2 error are reported as percentages; statistics pool input-layer-head-query-block allocation decisions.

Allocation Scope	Selection Signal	Mean Slots	Omitted Mass (%)		Output Error (%)	
			Mean ↓	P95 ↓	Mean ↓	P95 ↓
Position	Final reference mass	≈1,024	0.6012	0.9923	0.6231	4.237
Layer/Head/Position	Final reference mass	≈1,024	0.1721	0.6247	0.2817	0.9237
Input/Layer/Head/Position	Final reference mass	≈1,024	**0.0182**	**0.0664**	**0.0164**	**0.1013**
Input/Layer/Head/Position	**MALA online mass bound**	≈1,024	0.0188	0.0693	0.0174	0.1017

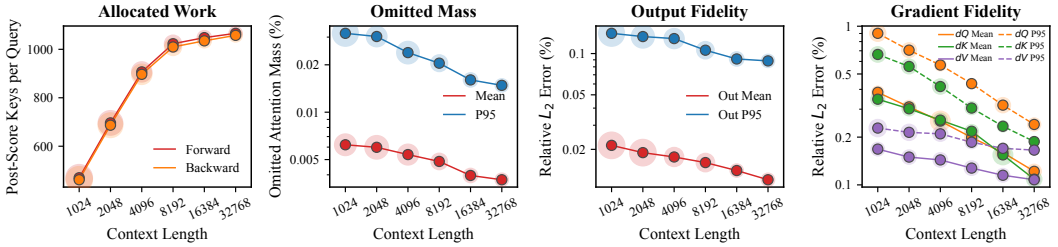

Figure 2: **Operator fidelity.** Forward and Backward allocated work is reported as mean post-score key slots per query. Actual omitted reference probability mass, relative output error, and relative gradient errors are measured over every query position against the same operator with screening disabled. Query positions are first aggregated within each sequence-layer-head state; reported means and P95 values pool these states, and marker halos reflect confidence intervals from sequence bootstrap.

training use 128 NVIDIA H100 GPUs; model-level evaluation and operator benchmarking use 8 H100 GPUs. Appendix C provides the work and cost accounting, baseline configurations, training schedules, hardware settings, and evaluation protocols.

Normalized-Mass Allocation at Matched Work. We isolate the value of allocating work from the realized attention distribution while holding total work fixed. At the evaluated 8K context length, all policies in Table 1 execute exactly the same total post-score work; their shared average of approximately 1,024 post-score key slots per query is induced by MALA under $\tau = 1$. We compare MALA's online decisions with three diagnostic reference-mass controls: a position-only allocation, a static layer-head-position allocation, and a per-instance allocation given the work realized by MALA for each decision. All three controls rank candidate regions using finalized per-instance reference mass; they differ only in how much work they assign to each decision. Static layer-head specialization improves over position-only allocation by capturing fixed specialization patterns. Compared with static layer-head-position allocation, per-instance allocation reduces mean omitted mass by $9.5\times$ and mean relative output L_2 error by $17.2\times$. Using only its evolving online state, MALA closely approaches this reference-mass oracle: mean omitted mass is 0.0188% versus 0.0182%, and mean relative output L_2 error is 0.0174% versus 0.0164%. At fixed post-score work, the reference-mass controls quantify the benefit of per-instance allocation, and MALA's online decisions recover nearly all of that benefit. Appendix C.2 details the control construction and exact work matching.

Operator Fidelity Across Context Lengths. We next test whether a single normalized-mass tolerance preserves the behavior of the reference attention operator as its context grows. For prefixes from 1K to 32K tokens, we run the standard screened operator and a screening-disabled reference on identical model states over the complete causal support. We measure realized post-score work, omitted reference probability mass, and relative L_2 errors in the output and in $\mathbf{dQ}$, $\mathbf{dK}$, and $\mathbf{dV}$. Figure 2 shows that, as context grows by $32\times$, mean forward work increases from 470 to 1,066 key slots per query and mean backward work from 462 to 1,058; beyond 8K, both remain between 1,010 and 1,066. This stable regime establishes 1,024 as the rounded empirical reference scale used below, induced by the canonical tolerance. Within this operator-fidelity suite and across all lengths, mean omitted probability mass is at most 0.0062% and its P95 is at most 0.032%; mean relative output L_2 error is at most 0.021% and its P95 at most 0.14%. Mean relative gradient errors are at most 0.38% for $\mathbf{dQ}$, 0.35% for $\mathbf{dK}$, and 0.17% for $\mathbf{dV}$; the corresponding worst P95 errors are 0.90%, 0.66%, and 0.23%. The stable work and low errors support the same tolerance across forward and backward execution. Appendix C.3 provides the data, sampling, gradient-probe, and aggregation protocols.

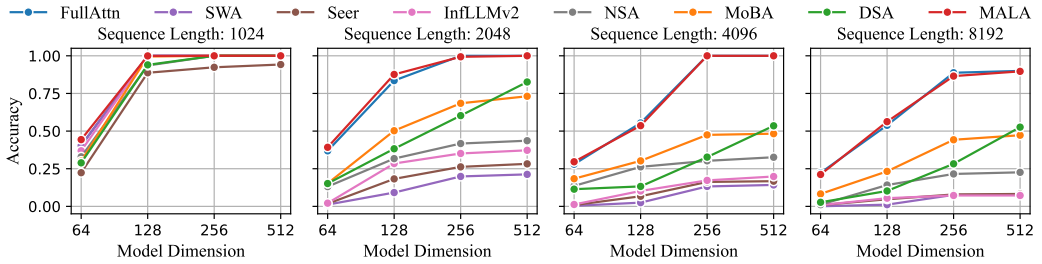

Figure 3: **Associative Recall**. Accuracy with 256 key-value pairs across sequence lengths and model dimensions. Fixed-budget sparse methods use a common 1,024-slot ceiling, with realized raw slots determined after causal clipping; MALA obtains its work from the canonical tolerance. The task directly tests the 256 bindings, with no injected distractors. MALA closely tracks FullAttn as the context grows, while the other sparse mechanisms lose a substantial fraction of the associations.

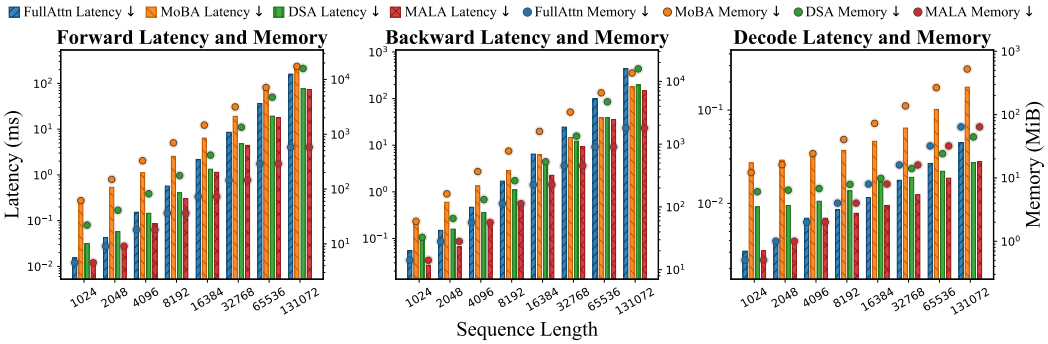

Figure 4: **Operator Latency and Memory**. End-to-end attention-operator latency (bars, left axes) and per-rank peak memory (markers, right axes) for the training forward and backward passes and autoregressive decoding with tensor parallelism $TP = 8$ on 8 H100 GPUs. Measurements include each method's complete score-discovery and selection path. MALA reduces latency while retaining FullAttn-level memory through in-kernel allocation.

Controlled Associative Recall. We use controlled associative recall (Arora et al., 2024) to test whether distribution-adaptive allocation preserves arbitrary long-range interactions before language priors and model-scale effects enter the evaluation. Guided by the approximately 1K scale observed above, we use raw post-score slots to control retained interactions: fixed-budget methods use a 1,024-slot ceiling after causal clipping, including DSA despite its larger value dimension, while MALA continues to use the canonical tolerance and realizes approximately the same work scale through normalized-mass allocation. We vary sequence length from 1,024 to 8,192 and d_{model} from 64 to 512. Figure 3 shows that MALA closely tracks FullAttn as the context grows. At sequence length 8,192 and $d_{model} = 512$, MALA reaches 89.67% accuracy and FullAttn reaches 89.97%, whereas DSA and MoBA reach 52.61% and 47.25%, respectively, and NSA reaches 22.61%. Thus attention-directed allocation preserves associations that fixed-support and externally selected alternatives miss under a common raw-slot reference. Appendix C.4 provides the data construction, training protocol, and work-matching details.

Operator Latency and Memory. Having established that normalized-mass allocation preserves operator behavior and long-range associations, we next measure whether it reduces the cost of complete attention operators. The sparse configurations target approximately 1,024 reference-equivalent post-score slots per query: MoBA uses 1,024 raw slots, DSA uses 256 because each slot contributes four times the reference PV work, and MALA continues to use $\tau = 1$ with distribution-determined work. This matching controls the work governed by the allocation decision; the benchmark (Tillet et al., 2019) measures the complete operator, including score discovery, method-specific allocation or selection, and data movement. All measurements use the matched $TP = 8$ configurations described in Appendix C.5. Figure 4 shows that allocation from the attention loop translates into consistent gains as the context grows. At 128K tokens, MALA reduces forward and backward latency during training by $2.2\times$ and $3.0\times$, and decoding latency during inference by $1.6\times$ relative to FullAttn. It provides the lowest forward and backward latency among the compared methods; its decoding latency is within 3% of DSA while remaining $1.6\times$ faster than FullAttn. Across all three operators and every

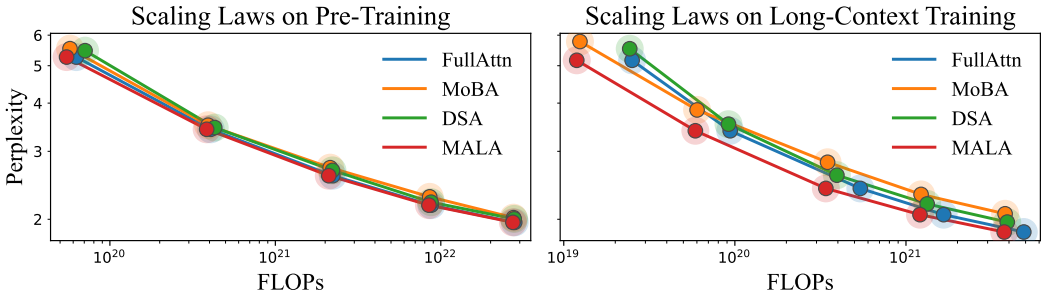

Figure 5: **Scaling Laws**. Perplexity as a function of total training FLOPs for models from 0.6B to 14B parameters during 4K pre-training (left) and 32K long-context training (right). The FLOP accounting includes complete score discovery and method-specific allocation or selection costs. MALA closely follows FullAttn in Perplexity while shifting the frontier toward lower training FLOPs.

Table 2: **Model-level evaluation summary.** Average knowledge and reasoning scores, together with RULER scores at the native 32K context length and after YaRN extrapolation to 128K. Detailed per-task results and uncertainty estimates are reported in Appendix D.

Model Scale	Method	Knowledge	Reasoning	RULER 32K	RULER 128K
14B	FullAttn	72.32	64.46	89.42	**65.84**
14B	MoBA	70.01	58.07	86.85	60.17
14B	DSA	70.11	62.04	87.59	63.05
14B	**MALA (ours)**	**72.48**	**64.66**	**89.45**	65.75
32B	FullAttn	75.62	75.67	92.70	82.03
32B	**MALA (ours)**	**75.75**	**76.10**	**92.71**	**82.56**

measured context length, its peak allocation matches FullAttn because allocation is fused into the attention loop and uses only standard attention state. These gains arise from avoiding low-contribution post-score execution; MALA continues to discover scores over the complete causal context.

Scaling Across Model Sizes. We next test whether normalized-mass allocation preserves language-model quality while reducing training computation across model sizes (Kaplan et al., 2020; Xiong et al., 2023). We study five model sizes from 0.6B to 14B under matched 4K pre-training and 32K long-context training configurations, using the same canonical tolerance for MALA and the same post-score FLOP-equivalent configurations as in the operator study. Figure 5 reports Perplexity against total training FLOPs, including parameterized model computation, complete score discovery, method-specific routing or indexing, and retained post-score execution. Across both stages and all model sizes, MALA closely tracks FullAttn in Perplexity while shifting the quality-compute frontier toward lower FLOPs. At 14B, its Perplexity differs from FullAttn by less than 0.001 after both pre-training and long-context training, while reducing the corresponding total FLOPs by 2.5% and 23.1%. The larger gain at 32K follows from allocating only the post-score work warranted by the realized distribution, while the complete causal QK score-discovery cost remains included. Detailed architectures, training schedules, allocation configurations, and FLOP accounting are provided in Appendix C.6.

Model-Level Evaluation. We finally test whether the preceding operator fidelity and scaling trends transfer to complete language models. We evaluate all four 14B attention variants from the scaling-law study and extend the FullAttn-MALA comparison to 32B models trained in a separate continued-training study. Table 2 shows that MALA remains comparable to FullAttn in average knowledge, reasoning, native 32K retrieval, and YaRN-extrapolated 128K retrieval at both scales. At 14B, MALA scores 72.48 and 64.66 on knowledge and reasoning versus 72.32 and 64.46 for FullAttn; at 32B, the corresponding averages are 75.75 and 76.10 versus 75.62 and 75.67. On native 32K RULER, MALA reaches 89.45 versus 89.42 at 14B and 92.71 versus 92.70 at 32B. Under YaRN (Peng et al., 2026) extrapolation to 128K, it reaches 65.75 versus 65.84 at 14B and 82.56 versus 82.03 at 32B. Bold and underlining mark the best and second-best results, respectively, within each model-scale and context-length group. Detailed per-task results and uncertainty estimates are reported in Appendix D. Together, these results show that attention-directed compute allocation preserves broad language-model capabilities and long-context behavior across model scales.

4 Related Work

MALA preserves complete causal score discovery and uses native softmax statistics to allocate post-score computation during training and inference; we relate this design to sparse attention, adaptive allocation and threshold calibration, hardware-aware sparse execution, and KV-cache management.

Sparse attention. Fixed sparse attention uses local windows, structured blocks, or global tokens (Child et al., 2019; Beltagy et al., 2020; Zaheer et al., 2020). Dynamic methods select tokens or blocks using retrieval scores, importance estimates, cache statistics, or learned routers (Tang et al., 2024; Lai et al., 2025; Li et al., 2024; Zhang et al., 2023; Xiao et al., 2024a; Qi et al., 2026; Zhao et al., 2025; Gao et al., 2024; Yuan et al., 2025b; Lu et al., 2025; DeepSeek-AI et al., 2025). By restricting the support before exact attention, these methods can avoid QK computation for excluded interactions. MALA retains complete causal QK score discovery and allocates post-score computation from the resulting scores and softmax state.

Adaptive allocation and threshold calibration. Twilight and Double-P adapt the retained computation through hierarchical top-p selection using estimated attention mass, with configurable cumulative-mass targets (Lin et al., 2025; Ni et al., 2026). Top-Theta instead calibrates thresholds offline for individual layers, heads, and sequence lengths (Berestizshevsky et al., 2026). MALA uses a shared normalized-contribution tolerance and derives allocation decisions directly from exact QK scores and native softmax normalizers inside the attention operator. The same tolerance applies across heads, layers, and inputs without separately calibrated thresholds, while their realized attention distributions determine the retained post-score work.

Hardware-aware sparse execution. FlashAttention reduces memory traffic through tiled execution and online softmax (Dao et al., 2022; Shah et al., 2024; Milakov & Gimelshein, 2018); FlexAttention and FlashInfer improve kernel programmability and serving efficiency (Dong et al., 2024; Ye et al., 2025). SpargeAttention combines a predicted attention map with an online softmax-aware filter (Zhang et al., 2025). BLASST compares block maxima with running maxima and calibrates length-dependent thresholds for prefill and decoding; it also explores sparsity-aware training (Yuan et al., 2025a). MALA tests scores against the accumulated or finalized softmax normalizer, using τ/L_q as a normalized-contribution tolerance. This rule supports a forward omitted-mass guarantee and paired fused forward and backward operators, with nested backward support derived from standard attention state and a shared tolerance across training, prefill, and decoding.

KV-cache management. KV-cache systems reduce inference cost by pruning, compressing, retrieving, or offloading cached states (Zhang et al., 2023; Li et al., 2024; Tang et al., 2024; Xiao et al., 2024a; Liu et al., 2024; Huang et al., 2025). MALA retains the full KV history for score discovery and value access, reducing post-score execution while leaving cache storage and placement as separate optimization opportunities.

5 Conclusion

We introduced MassAlloc Attention (MALA), a fused attention primitive that allocates post-score computation from normalized contributions under a shared tolerance. MALA preserves complete causal QK score discovery and pairs online forward allocation with backward allocation from the saved normalizer, deriving nested retained support using standard attention state. The same tolerance governs training, and inference. QK complexity remains quadratic, with savings arising from skipped post-score execution.

The matched-work allocation study shows that MALA closely approaches the per-instance reference-mass oracle under the same total post-score work. Operator-fidelity evaluations show low output and gradient errors across context lengths with a shared tolerance, while controlled associative recall shows that MALA closely tracks FullAttn as context grows. The tensor-parallel operator benchmarks show lower forward, backward and decoding attention latency while retaining FullAttn-level peak operator memory. The scaling-law experiments from 0.6B to 14B parameters show perplexity comparable to FullAttn with lower training FLOPs, and the model-level evaluations at 14B and 32B show comparable knowledge, reasoning, and long-context retrieval scores. These results indicate that allocating post-score computation according to normalized attention contributions can retain the evaluated capabilities of FullAttn while reducing attention computation.

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

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.

Federico Barbero, 'Alvaro Arroyo, Xiangming Gu, Christos Perivolaropoulos, Michael M. Bronstein, Petar Velivckovi 'c, and Razvan Pascanu. Why do llms attend to the first token? *ArXiv*, abs/2504.02732, 2025.

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

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

Konstantin Berestizshevsky, Renzo Andri, and Lukas Cavigelli. Top-theta attention: Sparsifying transformers by compensated thresholding, 2026. URL https://arxiv.org/abs/2502.08363.

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.

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

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.

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.

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.

Enrique Queipo de Llano, Alvaro Arroyo, Federico Barbero, Xiaowen Dong, Michael M. Bronstein, Yann LeCun, and Ravid Shwartz-Ziv. Attention sinks and compression valleys in llms are two sides of the same coin. *ArXiv*, abs/2510.06477, 2025.

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

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.

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.

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.

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.

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.

Xiangming Gu, Tianyu Pang, Chao Du, Qian Liu, Fengzhuo Zhang, Cunxiao Du, Ye Wang, and Min Lin. When attention sink emerges in language models: An empirical view. *ArXiv*, abs/2410.10781, 2024.

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.

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.

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*, 2021a.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021b. URL https://arxiv.org/abs/2103.03874.

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.

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.

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

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

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

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.

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.

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.

Chaofan Lin, Jiaming Tang, Shuo Yang, Hanshuo Wang, Tian Tang, Boyu Tian, Ion Stoica, Song Han, and Mingyu Gao. Twilight: Adaptive attention sparsity with hierarchical top-p pruning, 2025. URL https://arxiv.org/abs/2502.02770.

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*, 2024.

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

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.

Chenyang Lyu, Minghao Wu, and Alham Fikri Aji. Beyond probabilities: Unveiling the misalignment in evaluating large language models, 2024. URL https://arxiv.org/abs/2402.13887.

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.

Maxim Milakov and Natalia Gimelshein. Online normalizer calculation for softmax, 2018. URL https://arxiv.org/abs/1805.02867.

Wentao Ni, Kangqi Zhang, Zhongming Yu, Oren Nelson, Mingu Lee, Hong Cai, Fatih Porikli, Jongryool Kim, Zhijian Liu, and Jishen Zhao. Double-p: Hierarchical top-p sparse attention for long-context llms, 2026. URL https://arxiv.org/abs/2602.05191.

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

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*, pp. 1–22, 2023.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. Yarn: Efficient context window extension of large language models, 2026. URL https://arxiv.org/abs/2309.00071.

Yanlin Qi, Xinhang Chen, Huiqiang Jiang, Qitong Wang, Botao Peng, and Themis Palpanas. Pariskv: Fast and drift-robust kv-cache retrieval for long-context llms, 2026. URL https://arxiv.org/abs/2602.07721.

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 qa benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

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.

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.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

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.

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*, pp. 13003–13051, 2023.

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

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

Philippe Tillet, H. T. Kung, and David Cox. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pp. 10–19. Association for Computing Machinery, 2019.

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.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark, 2024. URL https://arxiv.org/abs/2406.01574.

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*, pp. 38–45, Online, October 2020. Association for Computational Linguistics.

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*, 2024a.

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. Duoattention: Efficient long-context llm inference with retrieval and streaming heads. *ArXiv*, abs/2410.10819, 2024b.

Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, et al. Effective long-context scaling of foundation models. *arXiv preprint arXiv:2309.16039*, 2023.

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.

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, 2025a. URL https://arxiv.org/abs/2512.12087.

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*, 2025b.

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.

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.

Jintao Zhang, Chendong Xiang, Haofeng Huang, Jia Wei, Haocheng Xi, Jun Zhu, and Jianfei Chen. Spargeattention: Accurate and training-free sparse attention accelerating any model inference, 2025. URL https://arxiv.org/abs/2502.18137.

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.

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.

Weilin Zhao, Zihan Zhou, Zhou Su, Chaojun Xiao, Yuxuan Li, Yanghao Li, Yudi Zhang, Weilun Zhao, Zhen Li, Yuxiang Huang, Ao Sun, Xu Han, and Zhiyuan Liu. Infllm-v2: Dense-sparse switchable attention for seamless short-to-long adaptation, 2025. URL https://arxiv.org/abs/2509.24663.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. Agieval: A human-centric benchmark for evaluating foundation models, 2023. URL https://arxiv.org/abs/2304.06364.

A Complete Forward and Backward Algorithms

Algorithms 1 and 2 give the complete tiled procedures corresponding to the online and offline allocation rules in Section 2.2.

Algorithm 1 MassAlloc Attention Forward

Require: Matrices $\mathbf{Q}, \mathbf{O} \in \mathbb{R}^{N_q \times d_h}$ and $\mathbf{K}, \mathbf{V} \in \mathbb{R}^{N_k \times d_h}$. Vector $\ell \in \mathbb{R}^{N_q}$. Thresh τ.
Divide $\mathbf{Q}, \mathbf{O}$ into T_r blocks of size B_q and $\mathbf{K}, \mathbf{V}$ into T_c blocks of size B_k.
for $1 \le i \le T_r$ **do**
  Initialize $\mathbf{O}_i = (0) \in \mathbb{R}^{B_q \times d_h}$, $\ell_i = (0) \in \mathbb{R}^{B_q}$, and $\mathbf{m}_i = (-\infty) \in \mathbb{R}^{B_q}$.
  Compute the per-row visible-key counts $\mathbf{L}_i$.
  Load $\mathbf{Q}_i$.
  for $T_c \ge j \ge 1$ **do**
    Load $\mathbf{K}_j$.
    Compute $\mathbf{S}_{ij} = \mathbf{Q}_i \mathbf{K}_j^T \in \mathbb{R}^{B_q \times B_k}$ and Apply causal mask to $\mathbf{S}_{ij}$ at boundaries.
    Compute $\tilde{m}_{ij} = \mathrm{rowmax}(\mathbf{S}_{ij}) \in \mathbb{R}^{B_q}$.
    if $\tilde{m}_{ij} - \mathbf{m}_i - \log(\ell_i) < \log(\tau / \mathbf{L}_i)$ for all rows **then**
      skip block (i, j) and **continue**.
    end if
    Load $\mathbf{V}_j$.
    Compute $\mathbf{m}_i^{\mathrm{new}} = \max(\mathbf{m}_i, \tilde{m}_{ij})$.
    Compute $\tilde{\mathbf{P}}_{ij} = \exp(\mathbf{S}_{ij} - \mathbf{m}_i^{\mathrm{new}}) \in \mathbb{R}^{B_q \times B_k}$.
    Compute $\tilde{\ell}_{ij} = \mathrm{rowsum}(\tilde{\mathbf{P}}_{ij}) \in \mathbb{R}^{B_q}$ and $\ell_i^{\mathrm{new}} = \exp(\mathbf{m}_i - \mathbf{m}_i^{\mathrm{new}})\ell_i + \tilde{\ell}_{ij} \in \mathbb{R}^{B_q}$.
    Update $\mathbf{O}_i \leftarrow \exp(\mathbf{m}_i - \mathbf{m}_i^{\mathrm{new}})\mathbf{O}_i + \tilde{\mathbf{P}}_{ij}\mathbf{V}_j \in \mathbb{R}^{B_q \times d_h}$.
    Update $\ell_i \leftarrow \ell_i^{\mathrm{new}}$ and $\mathbf{m}_i \leftarrow \mathbf{m}_i^{\mathrm{new}}$.
  end for
  Update $\mathbf{O}_i \leftarrow \mathrm{diag}(\ell_i)^{-1}\mathbf{O}_i$ and $\ell_i \leftarrow \mathbf{m}_i + \log(\ell_i)$.
  Store $\mathbf{O}_i, \ell_i$.
end for
Return $\mathbf{O}, \ell$.

Algorithm 2 MassAlloc Attention Backward

Require: Matrices $\mathbf{Q}, \mathbf{O}, \mathbf{dQ}, \mathbf{dO} \in \mathbb{R}^{N_q \times d_h}$ and $\mathbf{K}, \mathbf{V}, \mathbf{dK}, \mathbf{dV} \in \mathbb{R}^{N_k \times d_h}$. Vector $\ell \in \mathbb{R}^{N_q}$. Thresh τ.
Divide $\mathbf{Q}, \mathbf{O}, \mathbf{dQ}, \mathbf{dO}$ into T_r blocks of size B_q and $\mathbf{K}, \mathbf{V}, \mathbf{dK}, \mathbf{dV}$ into T_c blocks of size B_k.
for $1 \le j \le T_c$ **do**
  Initialize $\mathbf{dK}_j = (0) \in \mathbb{R}^{B_k \times d_h}, \mathbf{dV}_j = (0) \in \mathbb{R}^{B_k \times d_h}$.
  Load $\mathbf{K}_j, \mathbf{V}_j$.
  for $j \le i \le T_r$ **do**
    Compute the per-row visible-key counts $\mathbf{L}_i$.
    Load $\mathbf{Q}_i$.
    Compute $\mathbf{S}_{ji} = \mathbf{K}_j \mathbf{Q}_i^T \in \mathbb{R}^{B_k \times B_q}$ and Apply causal mask to $\mathbf{S}_{ji}$ at boundaries.
    if $\mathbf{S}_{ji} - \ell_i^\top < \log(\tau / \mathbf{L}_i)$ for all rows **then**
      skip block (j, i) and **continue**.
    end if
    Load $\mathbf{O}_i, \mathbf{dO}_i$.
    Compute $\mathbf{P}_{ji} = \exp(\mathbf{S}_{ji} - \ell_i^\top) \in \mathbb{R}^{B_k \times B_q}$ and $\mathbf{dP}_{ji} = \mathbf{V}_j \mathbf{dO}_i^\top \in \mathbb{R}^{B_k \times B_q}$.
    Compute $\mathbf{dS}_{ji} = \mathbf{P}_{ji} \circ (\mathbf{dP}_{ji} - \mathrm{rowsum}(\mathbf{O}_i \circ \mathbf{dO}_i)^\top) \in \mathbb{R}^{B_k \times B_q}$.
    Update $\mathbf{dV}_j \leftarrow \mathbf{dV}_j + \mathbf{P}_{ji}\mathbf{dO}_i$ and $\mathbf{dK}_j \leftarrow \mathbf{dK}_j + \mathbf{dS}_{ji}\mathbf{Q}_i$.
    Store $\mathbf{dQ}_i \leftarrow \mathbf{dQ}_i + \mathbf{dS}_{ji}^\top \mathbf{K}_j$.
  end for
  Store $\mathbf{dK}_j, \mathbf{dV}_j$.
end for
Return $\mathbf{dQ}, \mathbf{dK}, \mathbf{dV}$.

A.1 Fused Execution Details

Fused forward execution. The MALA decision is embedded in the ordinary tiled attention loop. Algorithm 1 parallelizes over query blocks. Within each query block, legal key blocks are visited in descending order, beginning with the diagonal. For each legal key block, the kernel first loads K and

computes QK. It loads V and executes the online-softmax and PV path only when the contribution test retains the tile. The running maximum, normalization sum, and output accumulator remain on chip exactly as in FlashAttention. After the traversal, the operator writes the output $\mathbf{O}$ and final ℓ required by backward. Allocation remains implicit in these standard attention states. The same fused forward operator serves training and inference prefill, with query blocks executing independently in both settings.

Fused backward execution. Algorithm 2 uses the standard key-major traversal so that each program can accumulate $\mathbf{dK}_j$ and $\mathbf{dV}_j$ locally while contributing to $\mathbf{dQ}_i$. For each key block, query blocks are visited in ascending order, beginning with the diagonal. QK is recomputed before the skip decision. For a retained tile, the kernel reconstructs its probability from the saved ℓ and executes the usual attention-backward equations; for a skipped tile, it saves the corresponding output-gradient load and value-dependent path. The log-sum-exp state ℓ saved by the forward pass also serves as the allocation state for backward, yielding a fused attention-backward primitive.

Fused decoding execution. Autoregressive decoding applies the same forward allocation rule with a short query and a longer KV sequence. The kernel evaluates every causal QK tile, then loads V and executes softmax and PV for retained tiles. All cached keys therefore remain score-accessible. Under split-KV decoding, each split initializes its own running maximum and normalization sum and tests tiles against a partial normalizer. This typically retains at least the work selected under the corresponding unsplit normalizer. The partial outputs are then combined by the standard online-softmax reduction. The persistent KV cache remains intact, and the decode-memory claim concerns the attention operator's working set.

Training and inference consistency. The same probability tolerance governs forward, backward, prefill, and decoding. Attention distributions and available normalizers determine the realized work in each setting. Backward can be more selective than forward because it tests against the final normalizer; this directly yields the support-nesting behavior described in Section 2.2.

B Forward Approximation Guarantees

We state the forward guarantee for a single query row; applying it independently to every valid row gives the result used in Section 2.2. Let s_k be the score of causally visible key k, and let R and O denote the sets of keys retained and omitted by MALA, respectively. Define the final retained and omitted unnormalized masses as

$$Z_R = \sum_{k \in R} \exp(s_k), \qquad Z_O = \sum_{k \in O} \exp(s_k). \tag{5}$$

Proposition 1 (Omitted probability mass)**.** *For a query with L_q causally visible keys, the forward rule in Equation 3 bounds the total probability mass assigned by dense softmax to omitted keys as*

$$\epsilon_q := \frac{Z_O}{Z_R + Z_O} \le \frac{\tau}{1+\tau}. \tag{6}$$

Proof. Consider an omitted key k and the retained normalizer Z_t accumulated when its tile is tested. The first legal tile is necessarily retained, so $Z_t > 0$ whenever a skip occurs. Equation 3 and the tile row maximum imply

$$\frac{\exp(s_k)}{Z_t} < \frac{\tau}{L_q}. \tag{7}$$

The final retained normalizer satisfies $Z_R \ge Z_t$, hence $\exp(s_k)/Z_R < \tau/L_q$. Summing over omitted keys and using $|O| \le L_q$ gives $Z_O/Z_R \le \tau$. Therefore

$$\epsilon_q = \frac{Z_O/Z_R}{1 + Z_O/Z_R} \le \frac{\tau}{1+\tau}. \tag{8}$$

□

Let p be the dense softmax distribution and let $\widetilde{p}$ be softmax renormalized over R, with zero probability on O. The retained probabilities gain total mass ϵ_q, while the omitted probabilities lose the same mass, so

$$\|p - \widetilde{p}\|_1 = 2\epsilon_q \le \frac{2\tau}{1+\tau}. \tag{9}$$

If $\|\boldsymbol{v}_k\| \leq V_{max}$ for every visible value, the dense output $\boldsymbol{o} = \sum_k p_k \boldsymbol{v}_k$ and the retained output $\widetilde{\boldsymbol{o}} = \sum_k \widetilde{p}_k \boldsymbol{v}_k$ consequently satisfy

$$\|\boldsymbol{o} - \widetilde{\boldsymbol{o}}\| \leq \sum_k |p_k - \widetilde{p}_k| \|\boldsymbol{v}_k\| \leq \frac{2\tau}{1+\tau} V_{\max}. \tag{10}$$

These statements assume exact arithmetic. In the fused implementation, the running normalizer and contribution test are evaluated in finite precision, so the formal bounds hold up to the corresponding rounding error. Section 3 measures omitted mass and output and gradient fidelity for the implemented operators directly.

C Experiment Settings

Within each experiment, attention variants use matched model scales, depth, hidden size, data, and optimization settings while retaining their native attention-specific configurations. Unless stated otherwise, MALA uses the canonical tolerance $\tau = 1$, with realized work determined separately for each layer, head, input, and sequence length. The matched-work and operator-fidelity studies use the same 14B MALA checkpoint obtained after the 32K long-context stage of the scaling-law study. Our software stack uses NVIDIA PyTorch container images (NVIDIA, 2022) and the Transformers framework (Wolf et al., 2020). Scaling-law (Kaplan et al., 2020; Xiong et al., 2023)and continued-training experiments use Megatron-LM (Shoeybi et al., 2019) on 128 NVIDIA H100 GPUs. Benchmark evaluation uses the EleutherAI LM Evaluation Harness (Gao et al., 2021) on 8 NVIDIA H100 GPUs. Operator benchmarking uses 8 H100 GPUs with tensor parallelism $TP = 8$.

C.1 Work and Cost Accounting

We distinguish three quantities throughout the experiments: realized allocation, architecture-normalized allocation, and complete system cost.

Mean post-score key slots per query. For method a, let $s_{a,q}$ be the number of key slots for which query row q executes the post-score path after QK score discovery. A slot is counted when it participates in an executed softmax/value/PV tile or sparse-attention kernel call. Consequently, the count follows the operator's actual tile or block granularity and includes aligned or masked slots when the kernel still executes their post-score arithmetic; positions removed before this path are not counted. We report

$$\bar{s}_a = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} s_{a,q}, \tag{11}$$

where $\mathcal{Q}$ contains all query-row instances included by the experiment, including the relevant examples, layers, heads, and positions. This realized mean differs from a nominal Top-K or block budget. In particular, causal clipping, block expansion, and tile sharing can change $\bar{s}_a$ even when two methods receive the same configured integer budget. We use this raw quantity for within-architecture comparisons, including the matched-work allocation study, to describe how much work MALA realizes at a fixed tolerance, and in associative recall to define a common retained-interaction reference across methods.

Post-score FLOP-equivalent work. Raw slot counts do not define a compute-matched comparison when attention variants use different value dimensions or numbers of QO heads. For forward attention, the dominant allocated matrix multiplication is PV, whose cost is $2h_a d_{v,a} \bar{s}_a$ FLOPs per query token, summed over all QO heads, when one multiply-accumulate counts as two FLOPs. Relative to a reference configuration with h_{ref} QO heads and value dimension $d_{v,\mathrm{ref}}$, we therefore define

$$\bar{s}_a^{\mathrm{eq}} = \bar{s}_a \frac{h_a d_{v,a}}{h_{\mathrm{ref}} d_{v,\mathrm{ref}}}. \tag{12}$$

We use the matched FullAttn/MALA head configuration as the reference within each experiment. Thus, when the number of QO heads is shared, one DSA slot with $d_v = 512$ is approximately four 128-dimensional reference slots. Softmax and elementwise operations are lower-order terms in this matching proxy and remain included in measured operator latency. We use this normalized quantity

when a cross-architecture comparison is intended to control allocated computation, specifically in the operator and scaling-law configurations. Associative recall instead matches raw mean post-score slots, because it controls the number of retained interactions rather than their architecture-dependent FLOPs.

Full operator cost. Post-score equivalence controls only the computation governed by the allocation decision. It excludes score discovery and method-specific selection. Full operator cost additionally includes MALA's complete causal QK score discovery, MoBA's pooling, routing, Top-K, index construction, rearrangement, and merging, and DSA's indexer preparation, quantization, dense index scores, Top-K, indices, and sparse MLA score computation. We use synchronized end-to-end operator latency and peak memory for the systems comparison. For scaling-law plots, we count these sequence-dependent terms together with the shared model-training term. This separation prevents a raw key count from hiding architectural differences and prevents post-score matching from being presented as an end-to-end compute match.

C.2 Normalized-Mass Allocation at Matched Work Setup

Throughout the matched-work and operator-fidelity studies, omitted mass denotes $1-\exp(\ell_{\text{screened}}-\ell_{\text{reference}})$, the probability mass assigned by the dense reference softmax to positions omitted by the screened operator. We report this fraction as a percentage in the experimental results.

We evaluate complete causal attention on 256 language-modeling sequences of length 8,192, covering every query position in every layer and query head. The resulting evaluation contains 58,720,256 runtime allocation decisions. For each decision, a screening-disabled reference provides the probability mass of every candidate region. The diagnostic controls retain the highest-mass regions under this reference; MALA uses its actual online allocation and selection rule.

All policies execute exactly the same total number of post-score key slots, with a shared mean of approximately 1,024 slots per query induced by MALA's realized allocation. All three diagnostic controls use the finalized screening-disabled distribution to rank candidate regions by probability mass at the same granularity as MALA; they differ only in the number of slots assigned to each decision. The position-only control averages MALA's realized work across inputs, layers, and heads at each query position, producing one shared position-dependent allocation. The static layer-head-position control averages only across inputs, producing a separate allocation for each query position, layer, and head that remains fixed across sequences. The per-instance control uses the exact number of slots realized by MALA for each input, query position, layer, and head. Given an allocation, each control retains the highest-mass candidate regions. Data-independent integer rounding makes the two static allocations match MALA's total slot count exactly. The per-instance control therefore separates the value of instance-dependent work allocation from the approximation introduced by making the selection decision online. Because every row in this study uses the same attention representation and post-score kernel shape, equality in total slots is also equality in post-score FLOPs.

C.3 Operator Fidelity Across Context Lengths Setup

We randomly sample 256 sequences from a mixture of knowledge, reasoning, and retrieval data and evaluate prefixes of 1,024, 2,048, 4,096, 8,192, 16,384, and 32,768 tokens. At every layer and query position, we apply the standard screened operator and a screening-disabled reference to the same queries, keys, values, causal support, and kernel configuration. We record the realized forward and backward post-score key slots per query and recover the probability mass omitted by MALA from the two softmax normalizers.

We measure relative L_2 errors in the output and in $\mathbf{dQ}$, $\mathbf{dK}$, and $\mathbf{dV}$. Each relative error is the L_2 norm of the difference divided by the L_2 norm of the corresponding reference tensor, and is reported as a percentage. Backward measurements use a deterministic Rademacher probe for $\mathbf{dO}$, providing a reproducible direction through the complete attention backward operator. We aggregate complete operators at head level, yielding 409,600 query-head-layer measurements and 81,920 KV-head-layer measurements at each context length. Means and percentiles are computed over these measurements; confidence intervals use sequence-level bootstrap so that positions, layers, and heads from the same sequence remain clustered.

C.4 Associative Recall Setup

Following prior work (Arora et al., 2024), we evaluate associative recall with 256 key-value pairs. We consider sequence lengths of 1,024, 2,048, 4,096, and 8,192 and model dimensions $d_{model} \in \{64, 128, 256, 512\}$. The repeated key-value content is placed contiguously at the beginning of each sequence, occupying the first $2n_{\text{kv}}n_{\text{pass}}$ tokens, where $n_{\text{kv}} = 256$ and n_{pass} is the number of passes. We then sample n_{kv} query positions without replacement from the remaining positions according to $p(t) \propto t^{a-1}$, where t is the position within this suffix. We use the default $a = 0.01$, so queries occur more frequently near the beginning of the suffix and become progressively sparser toward its end. We do not insert random padding or additional noise tokens. This setup tests recall without adding a separate task of filtering random padding or noise. The dataset contains 250K training examples and 1K test examples, and all models are trained for 100 epochs.

FullAttn attends to the complete causal history. Fixed-budget sparse variants use a common nominal ceiling of 1,024 post-score key slots per query. SWA, Seer, InfLLMv2, NSA, MoBA, and DSA retain their native support or selection mechanisms; their realized means are computed after causal clipping and implementation-specific block expansion. MALA uses the canonical tolerance $\tau = 1$; the fused operator realizes $\bar{s}$ near the same empirical work scale. This comparison uses raw retained interactions as its reference, aligning the number of accessible associations across methods. DSA follows the same raw-slot ceiling even though its $d_v = 512$ representation makes each retained slot more expensive than a 128-dimensional MALA or MoBA slot.

Table 3: **Associative-recall allocation configurations.** Fixed-budget sparse methods use a common 1,024-slot ceiling, with realized work computed after causal clipping and native block expansion. The raw-slot comparison aligns retained interactions; DSA therefore uses the same slot ceiling despite its larger value dimension. MALA uses $\tau = 1$, with work determined by normalized-mass allocation.

Method	Allocation Rule	$d_v/d_{v,\text{ref}}$	Nominal Slot Ceiling	Raw-Slot Reference
FullAttn	Complete causal history	1×	Full	Full
SWA / Seer / InfLLMv2 / NSA	Native support or selector	1×	1,024	after causal clipping
MoBA	Native block routing	1×	1,024	after causal clipping
DSA	Token-level Top-K	4×	1,024	after causal clipping
MALA	Normalized-mass tolerance $\tau = 1$	1×	-	measured near reference

C.5 Operator Latency and Memory Setup

We benchmark causal attention at sequence lengths from 1,024 to 131,072 on 8 NVIDIA H100 GPUs with tensor parallelism $TP = 8$. All methods use batch size one. Unless a configuration runs out of memory, latency is averaged over 100 iterations after 25 warm-up iterations. Latency is the synchronized wall-clock time of the tensor-parallel group, and peak memory is the maximum per-rank allocation across the eight ranks. Table 4 gives the operator shapes and allocation rules. Sparse configurations target approximately 1,024 reference-equivalent forward post-score slots per query; MALA realizes its work from $\tau = 1$, while MoBA and DSA use their native block and token selectors. FullAttn processes the complete causal context.

The measurements cover the complete attention-operator path. MoBA includes representative construction, router logits, Top-K, index construction, rearrangement, sparse attention, and output merging. DSA includes BF16 indexer inputs, FP8 quantization, dense indexer logits, Top-K/indices, and sparse MLA. MALA includes complete causal QK score discovery, its fused contribution test, and the retained post-score path. Model-level QKV/output projections, MLP layers, optimizer updates, and pipeline communication are outside this module benchmark. Peak memory is measured over the same operator path and represents operator working memory; persistent serving-time KV-cache capacity remains unchanged. Because MALA allocation uses only standard attention state and retains all KV entries, its expected memory behavior is parity with the matched FullAttn operator.

[1]The FullAttn implementation is available at `github.com/Dao-AILab/flash-attention`.

[2]The MoBA implementation is available at `github.com/MoonshotAI/MoBA`.

[3]The DSA indexer implementation is available at `github.com/deepseek-ai/DeepGEMM`.

[4]The DSA implementation is available at `github.com/Dao-AILab/flash-attention/flash_attn/cute`.

Table 4: **Operator benchmark configurations.** The sparse methods target comparable forward post-score FLOP-equivalent work, while reported latency and memory include their complete score-discovery and selection paths.

METHOD	n_h	$n_{h_{kv}}$	d_{qk}	d_v	ALLOCATION	TARGET EQ. SLOTS	PRECISION
FullAttn[1]	64	8	128	128	Full causal context	Full	BF16
MoBA[2]	64	64	128	128	$B = 128$, native routing	≈1,024	BF16
DSA[3,4]	64	1	576	512	Top-$K = 256$, FP8 indexer	≈1,024	BF16/FP8
MALA	64	8	128	128	$\tau = 1$, fused allocation	measured ≈1,024	BF16

C.6 SCALING LAWS SETUP

The scaling-law (Kaplan et al., 2020; Xiong et al., 2023) experiments use SmolLMCorpus (Ben Allal et al., 2024), the Qwen3 tokenizer (Team, 2025), the AdamW optimizer (Loshchilov & Hutter, 2017), the WSD learning-rate schedule (Hägele et al., 2024), and the compute-optimal scaling principles (Li et al., 2025; Hoffmann et al., 2022). We use a fixed random seed of 42 for all scaling-law runs. We summarize the meaning of the columns in Table 5 and clarify which hyperparameters are used by each attention variant.

- **Params**: target model scale. Parameter counts are matched approximately because the attention projections differ across variants.
- **PT/LCT Tok**: total numbers of tokens used in pre-training (PT) and long-context training (LCT), respectively. Each entry reports the two values in the order `PT, LCT`.
- **Batch**: tokens per optimization step.
- **PT/LCT LR**: learning rates used during PT and LCT, respectively. Each entry reports the two values in the order `PT, LCT`.
- n_{layers}: number of Transformer layers.
- d_{model}: model hidden size.
- n_h: number of QO heads.
- $n_{h_{kv}}$: number of KV heads.
- d_{qk} and d_v: per-head dimensions of the query/key and value representations, respectively.
- **PS FLOPs/Slot**: forward post-score PV FLOPs contributed by one raw key slot, aggregated over all QO heads. Counting one multiply-accumulate as two FLOPs, this is $2hd_v$. It is the per-slot factor used by Equation 12, not the full attention cost.
- **PT Alloc** and **LCT Alloc**: allocation controls used during PT and LCT. FullAttn entries give the complete context length, MoBA and DSA entries give nominal raw selected-key budgets, and MALA entries give its normalized-mass tolerance.

For models from 0.6B to 14B, training proceeds in two stages. PT uses a sequence length of 4,096 and a WSD learning-rate schedule (Hägele et al., 2024) whose peak learning rate is reported in the PT entry of Table 5. The decay phase ends at the corresponding LCT learning rate, which is $0.1\times$ the PT peak. LCT initializes from the final PT checkpoint, increases sequence length to 32,768, and uses the reported constant LCT learning rate.

- **FullAttn**: standard causal scaled dot-product attention over the complete 4,096-token PT or 32,768-token LCT sequence.
- **MoBA**: block-routed attention with one KV head per QO head and a nominal 1,024-slot budget during both PT and LCT.
- **DSA**: sparse latent attention with one KV head, $d_{qk} = 576$, and $d_v = 512$. Its post-score PV cost per raw slot is four times that of MoBA or MALA, so we use a nominal budget of 256 during both PT and LCT.
- **MALA**: grouped-query attention with eight KV heads and the canonical tolerance $\tau = 1$ throughout PT and LCT. Normalized-mass allocation yields a realized mean near the 1,024-slot reference level.

Each method retains its native head configuration: FullAttn and MALA use eight KV heads, MoBA uses one KV head per QO head, and DSA uses a single latent KV head. The sparse variants approximately match forward post-score FLOP-equivalent work across these configurations. Full training FLOPs additionally include score discovery, routing, indexing, and Top-K costs as applicable. FullAttn attends to the complete context and serves as the dense reference.

Table 5: **Self-Attention Variants Scaling Laws Configurations.** Complete model, training, and attention configurations used in the scaling-law experiments. PS FLOPs/Slot describes allocated forward post-score work; the FLOP-axis additionally includes complete score discovery and method-specific selection.

Algos	Params	PT/LCT Tok	Batch	PT/LCT LR	n_{layers}	d_{model}	n_h	$n_{h_{kv}}$	d_{qk}	d_v	PS FLOPs/Slot	PT Alloc	LCT Alloc
FullAttn	≈0.6B	12B,1.5B	0.256M	1e-3,1e-4	28	1024	16	8	128	128	4096	4096	32768
MoBA	≈0.6B	12B,1.5B	0.256M	1e-3,1e-4	28	1024	16	16	128	128	4096	$K = 1024$	$K = 1024$
DSA	≈0.6B	12B,1.5B	0.256M	1e-3,1e-4	28	1024	16	1	576	512	16384	$K = 256$	$K = 256$
MALA	≈0.6B	12B,1.5B	0.256M	1e-3,1e-4	28	1024	16	8	128	128	4096	$\tau = 1$	$\tau = 1$
FullAttn	≈1.7B	34B,4B	0.512M	8e-4,8e-5	28	2048	16	8	128	128	4096	4096	32768
MoBA	≈1.7B	34B,4B	0.512M	8e-4,8e-5	28	2048	16	16	128	128	4096	$K = 1024$	$K = 1024$
DSA	≈1.7B	34B,4B	0.512M	8e-4,8e-5	28	2048	16	1	576	512	16384	$K = 256$	$K = 256$
MALA	≈1.7B	34B,4B	0.512M	8e-4,8e-5	28	2048	16	8	128	128	4096	$\tau = 1$	$\tau = 1$
FullAttn	≈4B	80B,10B	1M	6e-4,6e-5	32	2560	32	8	128	128	8192	4096	32768
MoBA	≈4B	80B,10B	1M	6e-4,6e-5	32	2560	32	32	128	128	8192	$K = 1024$	$K = 1024$
DSA	≈4B	80B,10B	1M	6e-4,6e-5	32	2560	32	1	576	512	32768	$K = 256$	$K = 256$
MALA	≈4B	80B,10B	1M	6e-4,6e-5	32	2560	32	8	128	128	8192	$\tau = 1$	$\tau = 1$
FullAttn	≈8B	164B,20B	1.6M	4e-4,4e-5	36	4096	32	8	128	128	8192	4096	32768
MoBA	≈8B	164B,20B	1.6M	4e-4,4e-5	36	4096	32	32	128	128	8192	$K = 1024$	$K = 1024$
DSA	≈8B	164B,20B	1.6M	4e-4,4e-5	36	4096	32	1	576	512	32768	$K = 256$	$K = 256$
MALA	≈8B	164B,20B	1.6M	4e-4,4e-5	36	4096	32	8	128	128	8192	$\tau = 1$	$\tau = 1$
FullAttn	≈14B	296B,36B	2M	3e-4,3e-5	40	5120	40	8	128	128	10240	4096	32768
MoBA	≈14B	296B,36B	2M	3e-4,3e-5	40	5120	40	40	128	128	10240	$K = 1024$	$K = 1024$
DSA	≈14B	296B,36B	2M	3e-4,3e-5	40	5120	40	1	576	512	40960	$K = 256$	$K = 256$
MALA	≈14B	296B,36B	2M	3e-4,3e-5	40	5120	40	8	128	128	10240	$\tau = 1$	$\tau = 1$

The FLOP-axis uses the cumulative training FLOPs reported by Megatron-LM and therefore covers the complete model computation. To expose the attention-specific contribution to this statistic, we additionally record the work executed by each attention operator. We count one multiply-accumulate as two FLOPs and include the QK recomputation used by attention backward. Let $a_n = (n+1)/2$ be the mean number of legal causal keys per query for a sequence of length n, and let $\bar{K} = K - K(K-1)/(2n)$ be the corresponding mean retained work after causally clipping a fixed budget $K \leq n$. The per-layer, per-token sequence-dependent attention terms are

$$C_{\text{FullAttn}} = 8a_n h d_{qk} + 6a_n h d_v, \tag{13}$$

$$C_{\text{MoBA}} = 8\bar{K} h d_{qk} + 6\bar{K} h d_v + C_{\text{router}}, \tag{14}$$

$$C_{\text{DSA}} = 8\bar{K} h d_{qk} + 6\bar{K} h d_v + C_{\text{indexer}}, \tag{15}$$

$$C_{MALA} = 4a_n h d_{qk} + 4\bar{s}_b h d_{qk} + 2\bar{s}_f h d_v + 4\bar{s}_b h d_v. \tag{16}$$

For MALA, $\bar{s}_f$ and $\bar{s}_b$ are the mean forward and backward post-score slots per query measured during training. The 4K stage records $(\bar{s}_f, \bar{s}_b) = (906.78, 896.74)$, and the 32K stage records $(1{,}065.95, 1{,}057.64)$. Its first term counts forward score discovery and backward QK recomputation over the complete causal region; the remaining terms count forward PV and the retained $\mathbf{dQ}$, $\mathbf{dK}$, $\mathbf{dP}$, and $\mathbf{dV}$ paths. C_{router} includes MoBA pooling, dense query-to-block routing scores, and Top-K comparisons. C_{indexer} includes DSA's dense lightning-indexer score path, index transforms and quantization arithmetic, and Top-K comparisons.

C.7 Sparse Adaptation via Continued Training

We study adaptation at 32B through long-context continued training, using FullAttn as the dense reference. Both models are initialized from the same Qwen3-32B checkpoint and trained at sequence length 32,768 for 64B tokens with a batch size of 4M tokens and a constant learning rate of 1×10^{-5}. Both configurations use 64 layers, hidden size 5,120, 64 QO heads, 8 KV heads, and $d_{qk} = d_v = 128$. FullAttn uses the complete causal context. MALA uses the canonical tolerance $\tau = 1$, retains complete causal QK score discovery, and records its distribution-determined post-score work.

C.8 Benchmark Evaluation

We evaluate two groups of models. At 14B, we compare FullAttn, MoBA, DSA, and MALA from the scaling-law study in Appendix C.6. At 32B, we compare the FullAttn and MALA models from the continued-training setup in Appendix C.7. For each task, we report the mean and standard deviation over five evaluation runs with seeds 0, 42, 233, 666, and 1234. The standard benchmark suite covers knowledge, reasoning, and retrieval through MMLU (Hendrycks et al., 2021a; Lyu et al., 2024), MMLU-Pro (Wang et al., 2024), BBH (Suzgun et al., 2023), HellaSwag (Zellers et al., 2019), OBQA (Mihaylov et al., 2018), WinoGrande (Sakaguchi et al., 2021), PIQA (Bisk et al., 2020), GSM8K (Cobbe et al., 2021), Hendrycks-Math (Hendrycks et al., 2021b), ARC-C (Clark et al., 2018), AGIEval (Zhong et al., 2023), GPQA-Diamond (Rein et al., 2023), and RULER (Hsieh et al., 2024). We evaluate RULER at the native 32K context length and, after YaRN position extrapolation, at 128K. Benchmark prompts, shot counts, decoding settings, seeds, and metric aggregation are held fixed across attention variants within each model scale.

D Detailed Model-Level Results

Tables 6, 7, and 8 report the per-task results summarized in Table 2.

Table 6: **Knowledge Benchmark Results**. Knowledge evaluation of the scaling-law models at 14B and the continued-training models at 32B. MALA produces averages comparable to FullAttn at both scales and remains close across individual tasks.

Model	MMLU Gen|5-shot	MMLU-Pro 5-shot	BBH CoT|3-shot	HellaSwag 5-shot	OBQA 5-shot	WinoGrande 5-shot	Avg
14B-Non-Thinking, temperature=0.6, top_k=20, top_p=0.95, max_gen_toks=32768, seed=[0, 42, 233, 666, 1234]							
FullAttn	**78.21±0.31**	**64.96±0.46**	**82.59±0.48**	**80.64±0.38**	49.4±2.17	78.16±1.14	72.32
MoBA	75.78±0.48	58.42±0.84	82.1±0.44	79.7±0.37	49.49±2.07	74.57±1.47	70.01
DSA	72.5±0.53	59.56±0.65	81.42±0.62	80.22±0.31	49.41±2.25	77.57±1.19	70.11
MALA (ours)	78.14±0.31	64.67±0.44	82.54±0.48	80.61±0.36	**50.0±2.17**	**78.93±1.13**	**72.48**
32B-Thinking, temperature=0.6, top_k=20, top_p=0.95, max_gen_toks=32768, seed=[0, 42, 233, 666, 1234]							
FullAttn	77.27±0.34	**68.57±0.41**	**88.39±0.54**	83.4±0.37	**55.0±2.23**	**81.14±1.1**	75.62
MALA (ours)	**77.37±0.33**	68.33±0.41	88.23±0.55	**84.69±0.36**	**55.0±2.23**	80.87±1.13	**75.75**

Knowledge performance. Table 6 shows that MALA retains knowledge benchmark performance close to FullAttn, with average scores of 72.48 versus 72.32 at 14B and 75.75 versus 75.62 at 32B. At 14B, the differences on MMLU, MMLU-Pro, BBH, and HellaSwag are all within 0.29 points. At 32B, the largest difference is a 1.29-point gain on HellaSwag, while all other differences remain within 0.27 points. The small aggregate gains therefore support comparable performance across the evaluated knowledge tasks rather than a uniform improvement.

Table 7: **Reasoning Benchmark Results**. Reasoning evaluation of the scaling-law models at 14B and the continued-training models at 32B. MALA matches FullAttn on average at 14B and 32B without a fixed retained-work budget.

Model	PIQA 5-shot	GSM8K CoT|4-shot	Hendrycks-Math CoT|4-shot	ARC-C 0-shot	AGIEval 0-shot	GPQA-Diamond CoT|3-shot	Avg
14B-Non-Thinking, temperature=0.6, top_k=20, top_p=0.95, max_gen_toks=32768, seed=[0, 42, 233, 666, 1234]							
FullAttn	**81.63±0.86**	90.32±0.93	**62.46±0.65**	**61.95±1.44**	**50.56±0.71**	39.87±3.47	64.46
MoBA	77.54±0.94	87.53±1.02	38.67±1.39	61.35±0.63	46.5±0.81	36.87±3.52	58.07
DSA	80.35±0.82	89.38±0.95	53.24±0.86	61.74±1.33	47.0±0.69	40.53±3.86	62.04
MALA (ours)	81.62±0.85	**90.83±0.9**	62.34±0.64	61.93±1.47	50.54±0.71	**40.67±3.32**	**64.66**
32B-Thinking, temperature=0.6, top_k=20, top_p=0.95, max_gen_toks=32768, seed=[0, 42, 233, 666, 1234]							
FullAttn	**85.97±0.88**	93.33±0.69	**94.94±0.82**	69.54±1.34	**54.18±0.68**	56.06±3.54	75.67
MALA (ours)	85.94±0.88	**95.45±0.57**	94.25±0.83	**69.8±1.34**	53.87±0.66	**57.29±2.89**	**76.1**

Reasoning performance. Table 7 shows comparable aggregate reasoning performance, with average scores of 64.66 versus 64.46 for FullAttn at 14B and 76.10 versus 75.67 at 32B. At 14B, MALA remains within 0.12 points of FullAttn on PIQA, Hendrycks-Math, ARC-C, and AGIEval, with higher scores on GSM8K and GPQA-Diamond. At 32B, gains on GSM8K and GPQA-Diamond

coexist with modest decreases on Hendrycks-Math and AGIEval. These results indicate that the reduction in post-score computation preserves the evaluated reasoning performance overall, with task-specific differences.

Table 8: **Retrieval Benchmark Results**. RULER evaluation at the native 32K context length and after YaRN extrapolation to 128K. MALA remains comparable to FullAttn across both model scales and context lengths.

MODEL	NIAH-S	NIAH-MK	NIAH-MQ	NIAH-MV	RULER-VT	RULER-CWE	RULER-FWE	RULER-QA	AVG
14B Native 32K Sequence Length, seed=[0, 42, 233, 666, 1234]									
FullAttn	**100±0**	**98.0±0.65**	**98.8±0.68**	**95.5±0.72**	**99.44±0.2**	**75.7±0.69**	91.67±0.71	56.28±1.2	89.42
MoBA	99.2±0.4	93.2±0.66	98.3±0.71	94.8±0.73	99.08±0.33	66.8±0.99	91.3±0.72	52.02±1.64	86.85
DSA	99.8±0.2	95.2±0.69	97.8±0.79	95.4±0.74	98.87±0.31	69.05±0.89	91.1±0.74	53.51±1.52	87.59
MALA (ours)	**100±0**	97.5±0.68	98.6±0.68	95.3±0.73	99.03±0.22	74.32±0.74	**93.76±0.59**	**57.05±1.18**	**89.45**
32B Native 32K Sequence Length, seed=[0, 42, 233, 666, 1234]									
FullAttn	**100±0**	**99.87±0.09**	99.8±0.08	**99.4±0.13**	99.64±0.18	**85.18±0.51**	94.53±0.53	63.2±0.93	92.7
MALA (ours)	**100±0**	99.53±0.11	**99.95±0.05**	99.25±0.19	**99.88±0.07**	83.3±0.54	**96.33±0.81**	**63.44±1.02**	**92.71**
14B YaRN 128K Sequence Length, seed=[0, 42, 233, 666, 1234]									
FullAttn	98.4±0.51	**55.2±1.64**	**88.25±0.69**	64.1±1.16	93.04±0.63	**5.76±0.46**	**84.26±0.79**	**37.73±2.15**	**65.84**
MoBA	92.2±0.83	51.6±1.85	80.3±0.87	62.05±1.3	90.48±0.71	0.02±0.06	79.93±0.95	24.78±2.63	60.17
DSA	96.4±0.62	52.7±1.73	83.6±0.81	**64.3±1.16**	92.48±0.62	4.8±0.52	77.93±1.16	32.19±2.03	63.05
MALA (ours)	**98.6±0.53**	54.6±1.67	88.2±0.69	64.2±1.14	**93.08±0.63**	5.51±0.45	84.17±0.81	37.62±2.12	65.75
32B YaRN 128K Sequence Length, seed=[0, 42, 233, 666, 1234]									
FullAttn	**99.8±0.12**	**86.4±1.35**	95.7±0.56	**88.0±0.79**	**98.76±0.22**	45.68±1.03	93.27±1.25	48.61±2.17	82.03
MALA (ours)	99.65±0.13	86.3±1.35	**95.9±0.66**	87.75±0.79	98.69±0.23	**45.8±1.04**	**96.33±0.81**	**50.03±1.97**	**82.56**

Long-context performance. Table 8 shows that MALA closely tracks FullAttn at both context lengths. At native 32K, their average scores differ by only 0.03 points at 14B and 0.01 points at 32B. Both methods obtain lower aggregate scores under YaRN extrapolation to 128K, but remain close to each other: MALA scores 65.75 versus 65.84 at 14B and 82.56 versus 82.03 at 32B. At 14B and 128K, every reported per-task difference is within 0.60 points; at 32B, the higher aggregate score includes gains on RULER-FWE and RULER-QA. These results support comparable long-context retrieval performance under the evaluated extrapolation setting.