MASaLA: Mass-Aware Fusion of Sparse and Low-rank Attention

Anonymous authors
Paper under double-blind review

Abstract

The quadratic computational cost of dense attention remains a central obstacle to training Transformers on long inputs. Efficient approximations typically restrict each query to a subset of keys or compress global interactions through low-rank and kernel-based representations. Sparse methods preserve sharp, localized interactions but may miss diffuse global structure, whereas low-rank methods capture broad mixing but can blur concentrated interactions. We propose MASaLA, a mass-aware fusion of sparse and low-rank attention operators that retains dense query, key, value, and output projections. A block-sparse branch computes exact attention over selected key blocks, while a low-rank branch summarizes diffuse global interactions. To account for the branches' different normalization masses, MASaLA uses mass-aware fusion that rescales the sparse contribution according to its estimated relative attention mass. The framework computes attention at subquadratic cost without materializing the full attention matrix. We further show that including each query's own block in the sparse support gives the hybrid operator the capacity to attain full rank regardless of the low-rank budget, overcoming the rank restriction of a purely low-rank approximation. We evaluate MASaLA on long-context classification benchmarks spanning 16K–64K tokens.

1 Introduction

Transformers have become the dominant architecture for language, vision, and multimodal modeling, largely because self-attention can adaptively mix information across all pairs of tokens (Vaswani et al., 2017; Devlin et al., 2019; Brown et al., 2020; Dosovitskiy et al., 2021; Touvron et al., 2023), with vision and multimodal Transformers extending the same mechanism across modalities. However, this pairwise interaction is also the main barrier to extending models to longer input lengths. For a sequence of length N, standard attention forms the score matrix

$$Z = QK^\top/\sqrt{d_h}, \qquad P = \mathrm{softmax}(Z), \tag{1}$$

where $P \in \mathbb{R}^{N\times N}$ is applied to the value matrix V. Equivalently, for a single query token i, dense attention computes

$$d_i = \sum_{j=1}^{N} \exp(z_{ij}), \qquad o_i = \frac{1}{d_i}\sum_{j=1}^{N} \exp(z_{ij})\, v_j. \tag{2}$$

Here d_i is the attention denominator, and it determines the scale on which the weighted value average is normalized. The memory and compute cost of this dense $N \times N$ operator grows quadratically with context length. Systems advances such as FlashAttention improve the IO efficiency and practical runtime of exact dense attention, but they do not change the fact that exact attention still represents all pairwise token interactions (Dao et al., 2022; Dao, 2024). Therefore, algorithmic approximations of the attention operator remain essential for training and serving models with substantially longer contexts.

A large body of work reduces the cost of attention by exploiting either sparsity or low-rank structure. Sparse-attention methods restrict each query to a subset of keys, using local windows, global tokens, random patterns, routing, clustering, hashing, or large-entry detection (Child et al., 2019; Beltagy et al., 2020; Zaheer et al., 2020; Kitaev et al., 2020; Roy et al., 2021; Daras et al., 2020; Jiang et al., 2024). These methods are especially appealing when the attention distribution is low-entropy, peaked,

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.

or dominated by a small number of relevant keys. In such regimes, most of the denominator in Equation (2) may be contributed by only a small fraction of the full key set, and computing the corresponding high-mass interactions exactly can preserve the sharp token-level behavior of dense attention. In contrast, low-rank, kernel, and sketch-based methods approximate the dense attention operator through compressed features or landmarks, enabling global mixing without explicitly constructing the full matrix (Wang et al., 2020; Katharopoulos et al., 2020; Choromanski et al., 2021; Peng et al., 2021; Xiong et al., 2021). These methods are attractive when attention is high-entropy or diffuse, so that information is spread over many tokens and global context is better represented by a compressed dense summary. However, either assumption alone can be brittle: sparse methods may miss weak but globally important interactions, while low-rank methods may smooth out sharp, high-mass token-token dependencies.

This complementarity suggests approximating attention as a combination of sparse and low-rank components—a view supported both by evidence that the two families excel in different regimes of the attention matrix (Chen et al., 2021) and by classical robust PCA, which represents a matrix as a low-rank term plus a sparse residual (Candès et al., 2011; Chandrasekaran et al., 2011). Combining these structures inside attention is nonetheless subtle: naively adding the two branches can double-count the selected interactions, and prior work resolves this by subtracting the low-rank estimate on the sparse support (Chen et al., 2021) (Section 2 discusses this construction and its trade-offs in more depth). We take a complementary position and treat the fusion rule itself as a design axis, studying a fusion that does not subtract the low-rank branch. A separate difficulty remains regardless of subtraction: two branches normalized over supports of very different denominator mass can produce outputs on different scales even when both are useful, so balancing them is a question of output-level fusion, not entrywise correctness.

We propose MASaLA: Mass-Aware Fusion of Sparse and Low-rank Attention. MASaLA is built around a simple principle: use a block-sparse branch for high-confidence interactions and a low-rank branch for global residual context. The block-sparse branch selects blocks of keys that are likely to be highly relevant to a given query and computes exact attention over the selected support. A block scorer can instantiate the selection, but additionally forces the query's own block, and then computes exact attention over the union of selected and forced blocks. The low-rank branch summarizes global key/value information in a compressed representation and produces a query-dependent global context vector without materializing all pairwise scores.

For a given query token i, MASaLA forms a fused output

$$O_i = g_{S,i}\, m_{S,i}\, O_{S,i} + g_{R,i}\, O_{R,i}, \tag{3}$$

where $O_{S,i}$ is the block-sparse-branch output, $O_{R,i}$ is the low-rank-branch output, and g_S, g_R are learned gates. Let $d_{S,i}$ denote the estimated denominator mass covered by the block-sparse branch, and let $d_{R,i}$ denote the denominator estimate associated with the low-rank branch. We introduce the mass-aware correction

$$m_{S,i} = \frac{d_{S,i}}{d_{S,i} + \lambda_i\, d_{R,i} + \varepsilon}, \tag{4}$$

where λ_i can be fixed or learned, and may be shared across tokens or vary per token. Several empirical studies of long-context attention and KV-cache compression observe that a small fraction of critical tokens or sparse attention patterns can dominate the attention outcome (Zhang et al., 2023; Xiao et al., 2024; Tang et al., 2024; Jiang et al., 2024). Directly adding a sparse output to a low-rank output can distort the scale of the true attention computation, especially when the two branches are normalized by denominators of different magnitudes. The role of $m_{S,i}$ is to rescale the sparse contribution according to its estimated denominator mass, so that the two outputs are fused on a more comparable scale. Thus, MASaLA does not merely add two attention approximations; it explicitly accounts for their relative normalization scales. Beyond this normalization role, Section 4 shows that the same forced own-block support that stabilizes the fusion also fixes the *rank capacity* of the combined operator: because the diagonal is always inside the block-sparse support, the hybrid operator can reach rank N almost surely, matching exact dense attention, for any low-rank budget.

We emphasize a distinction from sparse-plus-low-rank weight parameterizations (Li et al., 2023; Han et al., 2024; Mozaffari et al., 2025), discussed further in Section 2: MASaLA keeps the learned Transformer projections dense and instead approximates the induced $N \times N$ attention operator after Q, K, V have been formed, targeting the quadratic attention matrix that appears for every input

sequence rather than the parameter count of W_Q, W_K, W_V, W_O. Combining these two orthogonal directions is also an interesting future research path.

Our contributions are as follows. First, we formulate long-context attention as a block-sparse plus low-rank operator construction rather than as a decomposition of learned projection matrices. Second, we combine a block-sparse branch (forcing the query's own block) with a modified RACE Attention (Joshi et al., 2026) low-rank sketch branch, targeting both sharp local/similar-token interactions and diffuse global context. Third, we introduce mass-aware fusion through m_{sparse}, with optional learnable λ. Fourth, we prove a rank-capacity lemma (Section 4) showing that the forced own-block support supplies a perfect matching that gives the MASaLA operator the same algebraic rank capacity as exact dense attention, and that this full-rank property holds almost surely for the actual, data-dependent operator, whereas a rank-limited branch used alone cannot represent generic dense attention exactly.

2 Related Work

Efficient attention for long sequences. Efficient Transformers aim to overcome the quadratic cost of dense self-attention (Tay et al., 2021; 2022). FlashAttention uses IO-aware tiling to avoid materializing the full attention matrix in HBM, improving the speed and memory of exact attention (Dao et al., 2022); FlashAttention-2 further improves throughput via better parallelism and work partitioning (Dao, 2024). Neither changes which interactions are computed; MASaLA instead changes the attention operator itself to avoid constructing all pairwise interactions, and Appendix B shows the two are complementary at the kernel level. Sparse Transformer uses structured sparse patterns (Child et al., 2019); Longformer combines local (sliding-window) and global attention, and BigBird additionally adds random attention (Beltagy et al., 2020; Zaheer et al., 2020); Reformer uses LSH (Kitaev et al., 2020); and Routing Transformer and SMYRF use clustering or asymmetric hashing to group queries and keys into content clusters (Roy et al., 2021; Daras et al., 2020). More recent inference-time methods exploit dynamic sparse attention or query-aware critical-token selection (Zhang et al., 2023; Xiao et al., 2024; Tang et al., 2024; Jiang et al., 2024), further supporting the view that only part of the full attention matrix is necessary for many inputs, though sparse selection alone may miss diffuse background interactions across the sequence.

Low-rank, kernel, and sketch attention. A second family approximates dense attention with low-dimensional features. Linformer projects the sequence dimension, assuming approximately low-rank attention (Wang et al., 2020); Linear Transformer replaces softmax with a feature-map kernel, giving $\phi(Q)(\phi(K)^\top V)$ over $\phi(Q)(\phi(K)^\top \mathbf{1})$ (Katharopoulos et al., 2020); Performer uses positive random features for linear-complexity softmax approximation (Choromanski et al., 2021), and Random Feature Attention uses random Fourier features with an optional gating mechanism, covering sequence modeling more broadly than autoregressive generation alone (Peng et al., 2021); Nyströmformer uses landmark-based Nyström methods (Xiong et al., 2021); YOSO replaces softmax with the collision probability of a locality-sensitive hash, $(1 - \arccos(q^\top k)/\pi)^\tau$, rather than approximating it (Zeng et al., 2021); and RACE Attention similarly replaces softmax with a sharpened angular kernel, estimated by soft-LSH sketches built on the RACE sketch of Coleman & Shrivastava (2020), using L hash tables and P Gaussian hyperplanes with a temperature-scaled softmax over the $\{\pm 1\}^P$ hypercube corners (Joshi et al., 2026). These methods give global context at subquadratic or linear cost but can struggle to represent sharp, high-confidence entries of the attention distribution. MASaLA uses a modified RACE Attention branch for global coverage while delegating high-mass interactions to a block-sparse branch; our modification exposes the branch's denominator d_R for fusion and omits its optional ensembling.

Sparse plus low-rank attention. The idea that attention may need both sparse and low-rank structure relates closely to sparse-plus-low-rank matrix decomposition. Robust PCA decomposes a matrix into a low-rank component and a sparse component (Candès et al., 2011; Chandrasekaran et al., 2011). Scatterbrain applies this to attention, combining a low-rank random-feature approximation with a sparse LSH-selected correction so selected entries match the unnormalized exponential attention exactly while non-selected entries are approximated by the low-rank component (Chen et al., 2021) – an elegant entrywise estimator with unbiasedness and reduced variance, but one whose analysis is primarily entrywise and does not directly address how to balance branch outputs

normalized by different denominators. MASaLA instead studies a training-time architecture with two *separately* normalized branches: the main issue is not only entrywise unbiasedness but branch-scale balance, which the factor $m_{S,i}$ of Equation (4) is designed for.

Sparse plus low-rank parameterization of model weights. Sparse-plus-low-rank structure has also been used to parameterize, compress, or adapt learned network weights: LoRA represents fine-tuning updates with low-rank factors (Hu et al., 2022), with subsequent work on quantized, weight-decomposed, restarted, and gradient-projected low-rank variants (Dettmers et al., 2023; Liu et al., 2024; Lialin et al., 2024; Zhao et al., 2024). Other work explicitly combines sparsity with low-rank structure in parameter space: LoSparse (Li et al., 2023), SLoPe (Mozaffari et al., 2025), and SLTrain (Han et al., 2024) each parameterize weight matrices with learned low-rank and sparse components, and sparse low-rank adaptation studies incorporating sparsity into low-rank fine-tuning (Ding et al., 2023). This direction is orthogonal to MASaLA: we do not replace W_Q, W_K, W_V, or W_O by sparse-plus-low-rank factors, but approximate the input-dependent attention operator induced by dense projections, reducing the sequence-length bottleneck rather than the parameter count. In short, SL-style methods decompose learned matrices in parameter space, whereas MASaLA constructs a sparse-plus-low-rank approximation in attention space.

3 METHODOLOGY

We instantiate MASaLA inside each attention layer by replacing dense attention with a block-sparse–low-rank hybrid operator that avoids constructing the full $N \times N$ score matrix, using a modified RACE Attention (Joshi et al., 2026) branch for global, diffuse context and a block-sparse branch for peaked, high-mass query–key interactions.

We describe the method for one attention head; multi-head MASaLA applies the same procedure independently per head and concatenates the outputs, as usual. Let $Q, K, V \in \mathbb{R}^{N \times d_h}$. Where dense attention computes Equation (2) directly over all N^2 pairs, MASaLA instead computes two outputs, O_R and O_S, with denominator proxies d_R, d_S, and combines them via the mass-aware fusion rule of Equation (7) (Algorithm 1). This fusion rule is identical in the causal and non-causal settings – it takes the branch outputs and denominators as black-box inputs and never itself references positions or masking. What differs is entirely internal to the two branches: the causal case restricts each branch's support to non-future positions (past key blocks only, with a triangular mask on the own block, for block-sparse; a causal prefix scan in place of global bucket summaries, for RACE). Appendix A gives a non-causal and causal version of *each branch algorithm* (Algorithms 2–5); the fusion step (Algorithm 1) itself is unchanged in both settings.

3.1 LOW-RANK BRANCH: MODIFIED RACE ATTENTION

We build on RACE Attention (Joshi et al., 2026), which replaces the softmax kernel with a sharpened angular kernel estimated by soft-LSH sketches in the spirit of the RACE sketch of Coleman & Shrivastava (2020): each of L_s hash tables softly assigns queries and keys to the corners of a $\{\pm 1\}^\gamma$ hypercube via a temperature-scaled softmax, accumulates key/value statistics inside these buckets, and lets each query read from the resulting summaries, giving a linear-time global approximation to dense attention. Our modification is to expose the branch's denominator d_R (needed by the fusion rule) and to omit the optional multi-projection ensembling of the original method. Full pseudocode for the non-causal branch is given in Appendix A, Algorithm 2; the causal variant is Algorithm 4.

3.2 SPARSE BRANCH: BLOCK-SPARSE ATTENTION

The sparse branch preserves the sharp token-level interactions that the low-rank branch may smooth. We treat it as a *block-sparse attention* module, specified by four ingredients:

1. **Block partition.** The sequence is partitioned into $M = \lceil N/B \rceil$ contiguous blocks of a fixed size B. Fixing the block size (rather than the block count) keeps the exact work linear in N.
2. **Block scorer.** A scorer $S(\cdot)$ assigns, to each query block, a relevance score for every key block. Different scorers may instantiate S (e.g. summaries of the low-rank hash signatures, extreme-value min/max bounds, or mean-pooled key representations). MASaLA is agnostic to

Algorithm 1: MASaLA: Mass-Aware Fusion of Sparse and Low-rank Attention

Input: hidden states $X \in \mathbb{R}^{N\times d}$; projections W_Q, W_K, W_V, W_O; RACE parameters L_s, γ, α; block size B and block scorer S; gate network G_θ; coefficient rule $\lambda_i > 0$; numerical floor $\varepsilon > 0$.
Output: hybrid attention output $O \in \mathbb{R}^{N\times d}$.
Project to one attention head: $Q \leftarrow XW_Q,\ K \leftarrow XW_K,\ V \leftarrow XW_V$.
Run the low-rank branch: $(O_R, d_R) \leftarrow \mathrm{RACE}(Q, K, V; L_s, \gamma, \alpha, \varepsilon)$.
Run the block-sparse branch: $(O_S, d_S) \leftarrow \textsc{BlockSparse}(Q, K, V; B, S, \varepsilon)$.
Compute mass-aware multiplier: $m_{S,i} \leftarrow \dfrac{d_{S,i}}{d_{S,i} + \lambda_i d_{R,i} + \varepsilon}$, $i = 1, \ldots, N$.
Compute token-wise gates: $(g_S, g_R) \leftarrow \sigma(G_\theta(Q))$.
Fuse branch outputs: $O_{\mathrm{head}} \leftarrow g_S \odot m_S \odot O_S + g_R \odot O_R$.
return $O \leftarrow O_{\mathrm{head}} W_O$.

this choice: the scorer only ranks candidate key blocks. Unless otherwise noted, our experiments use a raw-key-space scorer: each key block is summarized by the sum of its key vectors, each query token is scored against every key block's summary by a plain dot product, and a query block's score against a key block is the maximum of these per-token scores over the query block, summed across attention heads.

3. **Forced block.** Regardless of the scorer, every query additionally attends to its own block (the block containing the query), which is always included in the support.
4. **Exact attention on the selected support.** The top-k scored blocks together with the forced blocks define the support Ω. The branch computes exact softmax attention over Ω, returning a normalized block-sparse output O_S and its denominator $d_S = \sum_{j\in\Omega} \exp(Q_i^\top K_j/\sqrt{d_h})$.

Because the selection is over contiguous blocks and the forced blocks are fixed, the branch never materializes the full $N \times N$ matrix and stays in contiguous memory. Full pseudocode for the non-causal branch is given in Appendix A, Algorithm 3; the causal variant is Algorithm 5.

3.3 Denominator-aware block-sparse–low-rank fusion

The two branches live on different normalization scales: the block-sparse branch's support may cover only part of the row denominator, while the low-rank branch gives global coverage but may smooth sharp interactions, so directly adding the two outputs can over-amplify the sparse branch.

MASaLA introduces a mass-aware sparse multiplier

$$m_{S,i} = \frac{d_{S,i}}{d_{S,i} + \lambda_i\, d_{R,i} + \varepsilon}, \tag{5}$$

where $\lambda_i > 0$ can be a fixed constant, a single learnable scalar shared across tokens, or (as used for the reported results) a learnable function of q_i that varies per token; Appendix D gives the exact parameterization. The final MASaLA output uses two learned gates,

$$(g_{S,i}, g_{R,i}) = \sigma(G_\theta(q_i)) \in (0,1)^2, \tag{6}$$

and the fused output for token i is

$$O_i = g_{S,i}\, m_{S,i}\, O_{S,i} + g_{R,i}\, O_{R,i}. \tag{7}$$

The factor $m_{S,i}$ scales the block-sparse branch according to its denominator mass relative to the low-rank branch, while the gates learn how much each branch should contribute to the final representation. In some tasks we also test an *ungated* variant that removes the learned gates while keeping the mass-aware multiplier, i.e. $O_i = m_{S,i}\, O_{S,i} + O_{R,i}$; this isolates the effect of the mass-aware rescaling from that of the gating.

4 RANK-CAPACITY THEORY OF THE BLOCK-SPARSE–LOW-RANK HYBRID

Because every query's own block is always included in its block-sparse support (Section 3.2), the block-sparse branch's attention matrix $\hat{A}^S$ always has full support on the diagonal, while the RACE branch's attention matrix $\hat{A}^R$ has rank at most $r = L_s 2^\gamma$ (Section 3.1). The ungated MASaLA operator (Section 3.3) is exactly $\hat{A}_{\text{ung}} = M_S \hat{A}^S + \hat{A}^R$ for a diagonal mass-aware weight $M_S = \text{diag}(m_S)$: a sparse-plus-low-rank matrix whose support and rank budget are fixed by the block scorer and the RACE configuration. Appendix C gives this operator's precise definition and the analogous, gated operator actually deployed alongside it.

Lemma 1 (Rank capacity of the ungated MASaLA operator)**.** *Let $\hat{A}_{\text{ung}}(Q, K)$ denote this operator as a function of the already-projected queries and keys, with all branch parameters (RACE hash planes, L_s, γ, α, and λ_i's parameterization) fixed. If (Q, K) has a distribution absolutely continuous with respect to Lebesgue measure, then*

$$\boxed{\Pr\left[\,\text{rank}\left(\hat{A}_{\text{ung}}(Q,K)\right) = N\right] = 1.} \tag{8}$$

With probability 1 over the actual queries and keys the network produces — not merely under an idealized free-parameter model — the ungated MASaLA operator matches exact dense attention's rank capacity, N, for *any* low-rank budget r: forcing each query's own block into its support already supplies a perfect matching, removing the rank ceiling a rank-r branch alone cannot avoid. This same capacity already holds for Block-Sparse alone; the lemma's role is to confirm that adding the low-rank branch costs MASaLA nothing algebraically. We stress that full rank is necessary only for *exact* representation, not for small approximation error. Appendix C gives the full statement (including the sufficient condition for absolute continuity and the gated operator), the auxiliary facts and analytic-witness argument the lemma builds on, and the complete proof.

5 EXPERIMENTAL SETUP

We evaluate MASaLA across classification tasks to test whether the block-sparse–low-rank attention structure is useful beyond a single modality or dataset. Since our main motivation is efficient long-context modeling, we prioritize settings with relatively long input sequences or large tokenized inputs whenever possible. The benchmark suite includes long-document text classification, sentiment classification, fine-grained image classification, and low-resolution image classification. For all tasks, we compare attention variants under the same training and evaluation settings, and we report top-1 accuracy. Full task-specific hyperparameters, including learning rates, batch sizes, sequence lengths, model sizes, and training schedules, are reported in Appendix D.

Datasets and tasks. We evaluate on text and image benchmarks: ArXiv for scientific text classification; IMDB for binary sentiment classification (Maas et al., 2011); CIFAR-10 as a 1024-token pixel sequence (LRA-Image); Food-101 with 101 food categories (Bossard et al., 2014); Flowers-102 with 102 flower categories (Nilsback & Zisserman, 2008); and Oxford-IIIT Pet with 37 pet breeds (Parkhi et al., 2012). For ArXiv classification, the model predicts the scientific subject class of each document from inputs up to 64K tokens. We additionally evaluate on Needle-in-a-Haystack (NIAH), a synthetic retrieval diagnostic in which a single relevant fact (the "needle") is inserted at a random position in an otherwise irrelevant, uninformative sequence (the "haystack"), and the model must retrieve it from a single query at the end of the sequence; we sweep the haystack length from 512 to 65,536 tokens to directly test length generalization and the ability to retain sharp, individual-token information at whatever length each attention mechanism's sparse or low-rank approximation may blur. These tasks span long text inputs, long tokenized image inputs (up to 16K tokens), a synthetic long-range retrieval probe, and shorter pixel- and token-level sequences, so that attention efficiency is tested across regimes where the sequence length ranges from a few hundred to tens of thousands of tokens.

Attention variants. We compare RACE as the low-rank baseline, BLOCK-SPARSE as the sparse baseline, MASALA as our proposed mass-aware sparse-plus-low-rank method (and its ungated variant), and SCATTERBRAIN (Chen et al., 2021) as a sparse-plus-low-rank hybrid baseline. Scatterbrain

Table 1: NIAH: trained at N=1024, tested across lengths 512–65,536.

Method	512	1024	2048	4096	8192	16384	32768	65536	Avg.
MASaLA	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**	98.8%	84.2%	97.9%
Block-Sparse	24.6%	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**	90.6%
RACE	11.0%	**100.0%**	**100.0%**	52.0%	13.4%	6.4%	4.8%	2.2%	36.2%
Performer	19.2%	**100.0%**	88.2%	87.6%	88.2%	87.4%	83.2%	83.4%	79.7%
H2O	98.8%	**100.0%**	99.4%	99.4%	98.6%	99.4%	OOM	OOM	**99.3%**
Scatterbrain	4.2%	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**	88.0%
BS+RACE	14.0%	**100.0%**	96.2%	98.0%	83.4%	78.0%	14.0%	16.0%	62.5%
Full	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%

Table 2: ArXiv classification, trained and tested separately at each length 8K–64K.

	ArXiv @ 8K		ArXiv @ 16K		ArXiv @ 32K		ArXiv @ 64K		Average	
Method	Train ↑	Test ↑	Train ↑	Test ↑	Train ↑	Test ↑	Train ↑	Test ↑	Train ↑	Test ↑
MASaLA (ungated)	**100%**	81.80%	**100%**	90.55%	**100%**	**96.08%**	**100%**	**99.54%**	**100%**	**91.99%**
MASaLA	**100%**	80.65%	**100%**	**91.02%**	**100%**	95.03%	**100%**	99.08%	**100%**	91.44%
Scatterbrain	**100%**	76.73%	**100%**	87.33%	**100%**	91.03%	**100%**	90.76%	**100%**	86.46%
RACE	**100%**	**82.72%**	**100%**	86.27%	**100%**	90.79%	**100%**	91.00%	**100%**	87.69%
Block-Sparse	**100%**	69.12%	**100%**	76.73%	99.97%	76.50%	99.97%	77.42%	99.99%	74.94%
Full	99.92%	78.60%	99.94%	81.47%	100%	89.24%	99.94%	90.10%	99.95%	84.85%

fuses its two branches by an entrywise correction that subtracts, on the sparse support, the low-rank estimate of the same entries before adding the two branches under a single shared normalizer:

$$O_i \;=\; m_i\, O_i^{\mathrm{sparse}} \;+\; (1-m_i)\, O_i^{\mathrm{lowrank}}, \qquad m_i \;=\; \frac{d_i^{\mathrm{sparse}}}{d_i^{\mathrm{sparse}} + d_i^{\mathrm{lowrank}}}, \tag{9}$$

where O^{lowrank} is the low-rank output *after* its contribution on the selected support has been subtracted out (so the selected entries are not double-counted), and m_i is the sparse branch's share of the combined denominator mass. We use Scatterbrain's own fusion rule of Equation (9), but instantiate its sparse and low-rank components with our block-sparse and RACE branches, so that all hybrids share the same underlying branches and differ only in how they are fused. We also include BS+RACE as an ablation that combines the block-sparse and RACE branches but sets $m_{\mathrm{sparse}} = 1$, removing the mass-aware rescaling. Finally, we report FULL attention (exact, quadratic-cost attention, computed via FlashAttention-2 (Dao, 2024)) as a reference point in every table; because it does not reduce the quadratic computation cost that motivates this work, it is shown for context but excluded from the bolding. In every table we bold the highest value in each column among the efficient methods.

Hardware and evaluation protocol. All experiments were run on a single NVIDIA RTX PRO 6000 Blackwell GPU (48 GB). Within each task, all attention variants use the same model, training budget, optimizer, and evaluation protocol. Unless otherwise stated, we report top-1 accuracy.

6 Efficiency: Complexity and Fused Kernel

Asymptotic cost. For a sequence of length N (the causal case has the same asymptotic form), suppressing batch-size and head constants, exact attention costs $\Theta(N^2)$ in both compute and memory. Excluding block scoring, each MASaLA branch is linear in N: writing s for the block-sparse branch's fixed per-query key budget and $S_R = L_s 2^\gamma$ for the RACE branch's bucket-feature width,

$$C_{\mathrm{MASaLA}} = M_{\mathrm{MASaLA}} = \Theta\big(N(s+S_R)\big). \tag{10}$$

The ablation BS+RACE has the same asymptotic cost, differing only in the fusion rule. Appendix E derives Equation (10) branch by branch and provides a numerical example at $N = 32{,}000$.

Realizing it in a fused kernel. Equation (10) counts interactions and is hardware-independent; realizing it as measured savings needs a kernel that never materializes what the analysis avoids. We implement MASaLA as a fused Triton kernel following FlashAttention's tiling principle (Dao et al., 2022; Dao, 2024) – keep intermediates in on-chip SRAM rather than HBM – extended in

Table 3: Long-context vision classification (non-causal), 16K tokens.

	Oxford-IIIT Pet @ 16K		Flowers-102 @ 16K		Food-101 @ 16K		Average	
Method	Train ↑	Test ↑	Train ↑	Test ↑	Train ↑	Test ↑	Train ↑	Test ↑
MASaLA	**91.55%**	**22.51%**	**97.84%**	42.45%	**76.43%**	**28.33%**	**88.61%**	**31.10%**
Block-Sparse	23.96%	12.65%	77.25%	37.25%	59.63%	19.03%	53.61%	22.98%
RACE	59.89%	16.13%	91.66%	35.49%	67.57%	24.00%	73.04%	25.21%
BS+RACE	76.76%	19.00%	95.98%	**42.65%**	76.05%	26.67%	82.93%	29.44%
Full	92.60%	24.66%	97.81%	48.13%	78.93%	29.96%	89.78%	34.25%

Table 4: Short-length tasks: CIFAR-10 (causal, 1024 tokens) and IMDB (512 tokens).

	CIFAR-10 @ 1024		IMDB @ 512		Average	
Method	Train ↑	Test ↑	Train ↑	Test ↑	Train ↑	Test ↑
MASaLA	**99.83%**	**49.11%**	**90.68%**	75.93%	**95.25%**	62.52%
Block-Sparse	98.70%	46.34%	70.45%	66.14%	84.58%	56.24%
RACE	98.64%	48.63%	89.93%	77.41%	94.28%	63.02%
BS+RACE	98.72%	48.43%	90.20%	**77.71%**	94.46%	**63.07%**
Scatterbrain	99.14%	47.51%	72.85%	66.23%	86.00%	56.87%
Full	99.78%	46.18%	89.54%	78.67%	94.66%	62.42%

two ways: the block-sparse branch's key loop iterates only over each query's selected blocks, so its cost scales with s rather than N; and the low-rank branch together with the mass-aware fusion epilogue is fused into the *same* kernel, so the low-rank output is produced and consumed in registers and never written to HBM. The backward pass exploits the fusion's row-wise gradient structure to recompute both branch outputs rather than storing either, so MASaLA retains FlashAttention's $O(N)$ activation-memory scaling with a constant-factor overhead from the low-rank feature tensors that implementation-level changes reduce to within roughly 10% of FlashAttention at the full-model level (Appendix B.6). Appendix B gives the full design, the precise correspondence to and departures from FlashAttention, the backward-pass argument, the memory accounting, and the precision requirement (bf16, not fp16) that the low-rank fusion imposes. Measured wall-clock and peak-memory results are reported in Appendix B.7.

7 Results and Discussion

Tables 1–4 compare MASaLA with block-sparse, low-rank, hybrid, and exact full-attention baselines across a synthetic retrieval probe, long-context text, long-context vision, and short-sequence settings; throughout, "MASaLA" denotes the gated variant (Equation (7)) unless a row is explicitly labeled ungated. We read the results through two lenses that separate what MASaLA's fusion contributes. The first is *whether the sparse branch carries a learned, mass-dependent weight*: BS+RACE uses MASaLA's two branches unchanged but fuses them as the plain sum $O_i = g_{S,i}\, O_{S,i} + g_{R,i}\, O_{R,i}$, i.e. it fixes $m_{S,i}{=}1$, so every gap between it and MASaLA is attributable to the mass-aware weighting alone. The second is *the fusion formulation itself*: Scatterbrain (Chen et al., 2021) combines its branches under a single shared normalizer, subtracting the low-rank estimate on the sparse support to avoid double counting—a formulation premised on an *unbiased* low-rank estimator of the attention kernel—whereas MASaLA normalizes the two branches separately and fuses them by $m_{S,i}$. **Long-context vision** (Table 3) is MASaLA's clearest win: best or tied-best among efficient methods on every column but one (Flowers-102 test, 42.65% vs. 42.45% for BS+RACE) and far ahead on average (31.1% vs. 29.4% for BS+RACE, 25.2% for RACE, 23.0% for Block-Sparse), closing much of the gap to exact attention (34.3%) at a fraction of its cost; this is the regime the hybrid targets, where sharp local structure and diffuse global context coexist. Under the first lens, the **NIAH** probe (Table 1) is decisive: MASaLA averages 97.9% against BS+RACE's 62.5%, with BS+RACE collapsing to 14–16% at 32K–64K where MASaLA stays near-perfect. We hypothesize the cause is adaptivity rather than a shift of weight toward the sparse branch (note $m_{S,i} \leq 1$ can only *down*-weight it): without a per-token, mass-dependent $m_{S,i}$ the model cannot reduce the sparse branch's contribution when its support misses most of the attention mass, and a fixed unit weight then corrupts single-needle retrieval. NIAH also exposes a genuine limitation we report rather than hide: MASaLA is perfect up to and beyond its $N{=}1024$ training length out to 16K but *degrades* at extreme extrapolation (98.8% at

32K, 84.2% at 64K) while pure Block-Sparse stays at 100%, so the low-rank branch can actively hurt exact long-range retrieval—plausibly as RACE's soft-hash buckets, tuned at $N{=}1024$, increasingly collide as N grows—a failure $m_{S,i}$ mitigates but does not fully suppress. At 512 tokens—below the $N{=}1024$ training length—MASaLA and Full attention generalize robustly and remain at 100%, while Block-Sparse, RACE, and Performer fall to near chance (24.6%, 11.0%, 19.2%). Under the second lens, Scatterbrain and MASaLA differ in formulation, and the comparison probes whether a fixed, single-normalizer fusion suits a *biased* low-rank branch: RACE's kernel is a bounded hash-collision probability, not an unbiased estimate of the unbounded exponential, so the unbiasedness Scatterbrain's formulation relies on does not hold. On **causal ArXiv** classification (Table 2) MASaLA is strongest overall, with the ungated variant reaching 92.0% average test accuracy and the gated variant 91.4%, above Scatterbrain (86.5%), pure RACE (87.7%), pure Block-Sparse (74.9%), and exact full attention (84.9%), the last plausibly reflecting the implicit regularization of the sparse–low-rank structure at long context. On NIAH the two lenses combine instructively: Scatterbrain (88.0% average) beats BS+RACE (62.5%) even though *both* lack a learned mass weight, showing that formulation alone matters, and its subtraction keeps it at 100% at 32K–64K where MASaLA slips; yet it falls to 4.2% at 512 tokens and trails MASaLA on average, and on the short tasks it reaches only 56.9% vs. MASaLA's 62.5%. Read together, a learned scale-aware fusion is what lets a biased low-rank approximator contribute rather than distort, while a formulation designed for unbiased estimators is not uniformly superior. Two observations on ArXiv complicate any simple ranking and are worth stating: the *ungated* variant beats the gated one on average, and pure RACE wins outright at 8K (82.7% vs. MASaLA's 80.7–81.8%), so no single fusion choice dominates—consistent with, not contrary to, our claim that the fusion rule is a task-dependent design axis. One caveat keeps the ArXiv numbers from being over-read: the sequences are built by class-balanced streaming packing, so longer sequences contain more same-class evidence, which could inflate long-length accuracy independently of genuine long-range modeling (we did not run the per-document control that would separate the two); this does not affect the MASaLA-vs-BS+RACE or gated-vs-ungated comparisons, which hold all other factors fixed. On the **short-length tasks** (Table 4) no efficient method is uniformly best: MASaLA leads clearly on CIFAR-10 (49.1% test) but trails RACE and BS+RACE on IMDB test accuracy (75.9% vs. 77.4% and 77.7%; exact attention: 78.7%), and on the CIFAR/IMDB average BS+RACE and RACE come out slightly ahead (63.1% and 63.0% vs. 62.5%). Since BS+RACE differs from MASaLA only in the fusion weight, this reinforces the first lens from the opposite direction: the mass-aware weighting is decisive in the long-context regime it was designed for but not uniformly advantageous at short lengths, where the branches' denominator masses are more comparable and, moreover, the efficiency motivation is weakest because exact attention is entirely feasible and in fact wins IMDB outright. The relevant property there is not that MASaLA wins but that it degrades gracefully to a competitive baseline rather than failing—an important trait for a general-purpose attention layer expected to run across heterogeneous workloads.

8 Conclusion and Future Work

We introduced MASaLA, which fuses an exact block-sparse branch with a low-rank RACE branch via a mass-aware multiplier m_{sparse} correcting for their differing denominator scales; forcing each query's own block into the support gives the hybrid exact attention's rank capacity for any low-rank budget (Section 4), and its fused kernel matches FlashAttention's $O(N)$ memory (Figure 1) while running several times faster at long context (Figure 2). Our central finding is that the formulation of a hybrid attention, and even its necessity, is task-dependent rather than well defined: m_{sparse} is pivotal on long-context retrieval (removing it in BS+RACE drops NIAH from 97.9% to 62.5%) yet BS+RACE wins on short tasks; ungated fusion beats gated on ArXiv; and the low-rank branch hurts exact retrieval while helping vision. The fusion rule is likewise contingent on its branches: Scatterbrain's single-normalizer rule presumes an *unbiased* low-rank estimator, but RACE's kernel is a bounded collision probability, so a learned scale-aware fusion is what lets a biased approximator help rather than distort. Since no fusion rule, weighting, or branch dominates across tasks, a fixed hybrid design is a premature commitment; this motivates studying which fusion structure suits which *head or layer*, and when a biased low-rank approximation aids generative models versus corrupts precise token access, ideally at autoregressive scale, alongside scaling the causal variant to LM pretraining, a fused causal kernel (Appendix B), and composition with weight-space methods (Hu et al., 2022; Han et al., 2024).

AI USE STATEMENT

We used generative AI tools to assist with software implementation and refactoring, experiment planning and diagnostics, figure and table generation, and drafting and editing parts of the manuscript. The initial research question and MASaLA construction originated with the authors, who made all final decisions concerning the method, theoretical claims, experimental protocol, inclusion or exclusion of evidence, and scientific conclusions. AI-assisted code was tested against explicit correctness checks, theoretical arguments were manually verified and revised, and every reported number was regenerated from raw run outputs using the released scripts. The authors reviewed and take responsibility for the complete submission.

ETHICS STATEMENT

This work studies an efficient attention mechanism, evaluated on publicly available, established text, image, and retrieval benchmarks (Section 5). It does not involve human subjects, private user data, or new data collection, and does not fine-tune or adapt any pretrained foundation model. MASaLA is a general-purpose component for reducing the computational and memory cost of self-attention and could, like any efficient long-context method, be incorporated into larger systems (e.g. language or vision models) whose downstream use raises its own considerations around bias, safety, and misuse; this work does not itself study or mitigate such downstream risks. Performance on the evaluated benchmarks should not be interpreted as evidence of safety, fairness, or suitability for high-stakes deployment.

REPRODUCIBILITY STATEMENT

Section 3 specifies the MASaLA parameterization, Appendix D gives model initialization and per-task training and evaluation configurations, and Section 5 together with Appendix A describe the models and tasks. Complete proofs and assumptions are provided in Appendix C. The supplementary material includes the implementation code, configuration files, frozen run manifest, exact trainable parameter counts, validity and exclusion rules, raw per-run result records, and scripts that regenerate the tables and figures.

REFERENCES

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

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision (ECCV)*, volume 8694 of *Lecture Notes in Computer Science*, pp. 446–461. Springer, 2014. doi: 10.1007/978-3-319-10599-4_29.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901, 2020.

Emmanuel J. Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM*, 58(3):11:1–11:37, 2011. doi: 10.1145/1970392.1970395.

Venkat Chandrasekaran, Sujay Sanghavi, Pablo A. Parrilo, and Alan S. Willsky. Rank-sparsity incoherence for matrix decomposition. *SIAM Journal on Optimization*, 21(2):572–596, 2011. doi: 10.1137/090761793.

Beidi Chen, Tri Dao, Eric Winsor, Zhao Song, Atri Rudra, and Christopher Ré. Scatterbrain: Unifying sparse and low-rank attention. In *Advances in Neural Information Processing Systems*, volume 34, pp. 17413–17426, 2021.

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

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Łukasz Kaiser, et al. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021.

Benjamin Coleman and Anshumali Shrivastava. Sub-linear RACE sketches for approximate kernel density estimation on streaming data. In *Proceedings of The Web Conference (WWW)*, pp. 1739–1749, 2020. doi: 10.1145/3366423.3380244.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations*, 2024.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, volume 35, pp. 16344–16359, 2022.

Giannis Daras, Nikita Kitaev, Augustus Odena, and Alexandros G. Dimakis. SMYRF: Efficient attention using asymmetric clustering. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6476–6489, 2020.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems*, volume 36, pp. 10088–10115, 2023.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pp. 4171–4186, 2019. doi: 10.18653/v1/N19-1423.

Ning Ding, Xingtai Lv, Qiaosen Wang, Yulin Chen, Bowen Zhou, Zhiyuan Liu, and Maosong Sun. Sparse low-rank adaptation of pre-trained language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4133–4145, 2023. doi: 10.18653/v1/2023.emnlp-main.252.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936. doi: 10.1007/BF02288367.

Andi Han, Jiaxiang Li, Wei Huang, Mingyi Hong, Akiko Takeda, Pratik Jawanpuria, and Bamdev Mishra. SLTrain: A sparse plus low-rank approach for parameter and memory efficient pretraining. In *Advances in Neural Information Processing Systems*, volume 37, pp. 118267–118295, 2024.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. MInference 1.0: Accelerating pre-filling for long-context LLMs via dynamic sparse attention. In *Advances in Neural Information Processing Systems*, volume 37, pp. 52481–52515, 2024.

Sahil Joshi, Agniva Chowdhury, Amar Kanakamedala, Ekam Singh, Evan Tu, and Anshumali Shrivastava. RACE attention: A strictly linear-time attention layer for training on outrageously large contexts. In *International Conference on Learning Representations*, 2026. arXiv:2510.04008.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *PMLR*, pp. 5156–5165, 2020.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020.

Yixiao Li, Yifan Yu, Qingru Zhang, Chen Liang, Pengcheng He, Weizhu Chen, and Tuo Zhao. LoSparse: Structured compression of large language models based on low-rank and sparse approximation. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *PMLR*, pp. 20336–20350, 2023.

Vladislav Lialin, Sherin Muckatira, Namrata Shivagunde, and Anna Rumshisky. ReLoRA: High-rank training through low-rank updates. In *International Conference on Learning Representations*, 2024.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-decomposed low-rank adaptation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *PMLR*, pp. 32100–32121, 2024.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, 2011.

Leon Mirsky. Symmetric gauge functions and unitarily invariant norms. *The Quarterly Journal of Mathematics*, 11(1):50–59, 1960. doi: 10.1093/qmath/11.1.50.

Boris S. Mityagin. The zero set of a real analytic function. *Mathematical Notes*, 107(3):529–530, 2020. doi: 10.1134/S0001434620030189.

Mohammad Mozaffari, Amir Yazdanbakhsh, Zhao Zhang, and Maryam Mehri Dehnavi. SLoPe: Double-pruned sparse plus lazy low-rank adapter pretraining of LLMs. In *International Conference on Learning Representations*, 2025.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP)*, pp. 722–729, 2008. doi: 10.1109/ICVGIP.2008.47.

Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3498–3505, 2012. doi: 10.1109/CVPR.2012.6248092.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A. Smith, and Lingpeng Kong. Random feature attention. In *International Conference on Learning Representations*, 2021.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021. doi: 10.1162/tacl_a_00353.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. QUEST: Query-aware sparsity for efficient long-context LLM inference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *PMLR*, pp. 47901–47911, 2024.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long Range Arena: A benchmark for efficient transformers. In *International Conference on Learning Representations*, 2021.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *ACM Computing Surveys*, 55(6):109:1–109:28, 2022. doi: 10.1145/3530811.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, 2024.

Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A Nyström-based algorithm for approximating self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 14138–14148, 2021.

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. In *Advances in Neural Information Processing Systems*, volume 33, pp. 17283–17297, 2020.

Zhanpeng Zeng, Yunyang Xiong, Sathya N. Ravi, Shailesh Acharya, Glenn Fung, and Vikas Singh. You only sample (almost) once: Linear cost self-attention via Bernoulli sampling. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *PMLR*, pp. 12321–12332, 2021.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H_2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 34661–34710, 2023.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. GaLore: Memory-efficient LLM training by gradient low-rank projection. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *PMLR*, pp. 61121–61143, 2024.

A BRANCH ALGORITHMS FOR MASALA

This appendix gives the branch-level pseudocode used by Algorithm 1: Algorithm 2 (non-causal RACE) and Algorithm 3 (non-causal block-sparse) implement the branches as described in Sections 3.1–3.2. Algorithms 4 and 5 give the causal variant of each branch, obtained by restricting that branch alone to non-future positions, as described in Section 3. The fusion step, Algorithm 1, is *not* restated for the causal case: it is used identically, unmodified, in both settings, since it consumes only the branch outputs O_S, O_R and denominators d_S, d_R and never itself refers to token positions.

Algorithm 2: RACE low-rank attention branch (non-causal)

Input: $Q, K, V \in \mathbb{R}^{N\times d_h}$; number of hash tables L_s; number of hyperplanes γ; temperature $\alpha > 0$; numerical floor $\varepsilon > 0$.
Output: low-rank output $O_R \in \mathbb{R}^{N\times d_h}$; denominator proxy $d_R \in \mathbb{R}^N$.
Set $R \leftarrow 2^\gamma$ and $\mathcal{V} \leftarrow \{\pm 1\}^\gamma$.
for $\ell = 1, \dots, L_s$ **do**
  Draw $W^{(\ell)} \in \mathbb{R}^{\gamma\times d_h}$ with i.i.d. Gaussian rows.
  Build $\Phi_Q^{(\ell)}, \Phi_K^{(\ell)} \in \mathbb{R}^{N\times R}$ with rows $[\phi^{(\ell)}(x)]_r = \dfrac{\exp\{\alpha \tanh(W^{(\ell)}x)^\top v_r\}}{\sum_{r'=1}^{R}\exp\{\alpha \tanh(W^{(\ell)}x)^\top v_{r'}\}}$, $x \in \{Q_i, K_j\}$.
  Compute bucket mass and value summaries *over all N keys*: $A^{(\ell)} \leftarrow (\Phi_K^{(\ell)})^\top \mathbf{1}_N$, $B^{(\ell)} \leftarrow (\Phi_K^{(\ell)})^\top V$.
Compute table-averaged numerator and denominator: $\mathrm{Num} \leftarrow \frac{1}{L_s}\sum_{\ell=1}^{L_s}\Phi_Q^{(\ell)}B^{(\ell)}$, $\mathrm{Den} \leftarrow \frac{1}{L_s}\sum_{\ell=1}^{L_s}\Phi_Q^{(\ell)}A^{(\ell)}$.
return $O_R \leftarrow \mathrm{diag}(\mathrm{Den}+\varepsilon)^{-1}\mathrm{Num}$, $d_R \leftarrow \mathrm{Den}$.

Algorithm 3: Block-sparse exact attention branch (non-causal)

Input: $Q, K, V \in \mathbb{R}^{N\times d_h}$; block size B; block scorer $S(\cdot)$; number of selected blocks k; numerical floor $\varepsilon > 0$.
Output: block-sparse output $O_S \in \mathbb{R}^{N\times d_h}$; denominator $d_S \in \mathbb{R}^N$.
Partition $[N]$ into $M = \lceil N/B \rceil$ contiguous blocks $\{\mathcal{B}_1, \dots, \mathcal{B}_M\}$.
for *each query block* c **do**
  Score *all* candidate key blocks: $\mathrm{score}(c, c') \leftarrow S(\mathcal{B}_c, \mathcal{B}_{c'})$ for all c'.
  Select the top-k key blocks by score; call this set $\mathcal{T}_c$.
  Form the support $\Omega_c \leftarrow \mathcal{T}_c \cup \{\text{own block } c\}$.
  Compute exact block scores $Z_c \leftarrow Q[\mathcal{B}_c]\, K[\Omega_c]^\top/\sqrt{d_h}$.
  Compute weights and denominators $A_c \leftarrow \exp(Z_c)$, $d_S[\mathcal{B}_c] \leftarrow A_c\mathbf{1}$.
  Compute normalized output $O_S[\mathcal{B}_c] \leftarrow \mathrm{diag}(d_S[\mathcal{B}_c]+\varepsilon)^{-1}A_c\, V[\Omega_c]$.
return O_S, d_S.

B FUSED KERNEL: DESIGN, RELATION TO FLASHATTENTION, AND BACKWARD PASS

This appendix expands Section 6. We describe what the MASaLA kernel inherits from FlashAttention, the two places it departs, the backward pass that keeps its memory linear, and the precision constraint the low-rank fusion imposes. Throughout, N is the sequence length, d_h the head dimension, B the block size, s the per-query key budget (selected plus forced blocks, times B), L_s the number of RACE tables, and $R = 2^\gamma$ the buckets per table, so $S_R = L_s R$ is the RACE feature width.

Algorithm 4: RACE low-rank attention branch (causal)

Input: $Q, K, V \in \mathbb{R}^{N \times d_h}$; number of hash tables L_s; number of hyperplanes γ; temperature $\alpha > 0$; chunk size C; numerical floor $\varepsilon > 0$.
Output: causal low-rank output $O_R \in \mathbb{R}^{N \times d_h}$; denominator proxy $d_R \in \mathbb{R}^N$.
Set $R \leftarrow 2^\gamma$; partition $[N]$ into $n = \lceil N/C \rceil$ contiguous chunks $\{\mathcal{C}_1, \dots, \mathcal{C}_n\}$.
for $\ell = 1, \dots, L_s$ **do**
  Draw $W^{(\ell)}$ and build $\Phi_Q^{(\ell)}, \Phi_K^{(\ell)} \in \mathbb{R}^{N \times R}$ exactly as in Algorithm 2.
  for *each chunk* $c = 1, \dots, n$ **do**
    Compute chunk-local mass and value sums: $a_c^{(\ell)} \leftarrow \sum_{j \in \mathcal{C}_c} \Phi_{K\,j}^{(\ell)}$,
    $b_c^{(\ell)} \leftarrow \sum_{j \in \mathcal{C}_c} \Phi_{K\,j}^{(\ell)} \otimes V_j$.
  Compute the *exclusive* causal prefix state over chunks: $A_c^{(\ell)} \leftarrow \sum_{c' < c} a_{c'}^{(\ell)}$,
  $B_c^{(\ell)} \leftarrow \sum_{c' < c} b_{c'}^{(\ell)}$ (i.e. mass/value from all strictly-earlier chunks only).
for *each query* i *in chunk* c **do**
  Inter-chunk term (exact contribution of all strictly-earlier chunks):
  $\mathrm{num}_i^{\mathrm{inter}} \leftarrow \frac{1}{L_s} \sum_\ell \Phi_{Q\,i}^{(\ell)} B_c^{(\ell)}$,
  $\mathrm{den}_i^{\mathrm{inter}} \leftarrow \frac{1}{L_s} \sum_\ell \Phi_{Q\,i}^{(\ell)} \cdot A_c^{(\ell)}$.
  Intra-chunk term (exact, causally masked contribution of chunk c itself):
  $M_{ij} \leftarrow \frac{1}{L_s} \sum_\ell \Phi_{Q\,i}^{(\ell)} \cdot \Phi_{K\,j}^{(\ell)}$ for $j \in \mathcal{C}_c$ with $j \le i$, else 0;
  $\mathrm{num}_i^{\mathrm{intra}} \leftarrow \sum_{j \in \mathcal{C}_c} M_{ij} V_j$,
  $\mathrm{den}_i^{\mathrm{intra}} \leftarrow \sum_{j \in \mathcal{C}_c} M_{ij}$.
  Combine: $\mathrm{Num}_i \leftarrow \mathrm{num}_i^{\mathrm{inter}} + \mathrm{num}_i^{\mathrm{intra}}$,
  $\mathrm{Den}_i \leftarrow \mathrm{den}_i^{\mathrm{inter}} + \mathrm{den}_i^{\mathrm{intra}}$.
return $O_R \leftarrow \mathrm{diag}(\mathrm{Den} + \varepsilon)^{-1} \mathrm{Num}$,
$d_R \leftarrow \mathrm{Den}$.

Algorithm 5: Block-sparse exact attention branch (causal)

Input: $Q, K, V \in \mathbb{R}^{N \times d_h}$; block size B; causal block scorer $S(\cdot)$; number of selected blocks k; numerical floor $\varepsilon > 0$.
Output: causal block-sparse output $O_S \in \mathbb{R}^{N \times d_h}$; denominator $d_S \in \mathbb{R}^N$.
Partition $[N]$ into $M = \lceil N/B \rceil$ contiguous blocks $\{\mathcal{B}_1, \dots, \mathcal{B}_M\}$.
for *each query block* c **do**
  Score only *past* key blocks: $\mathrm{score}(c, c') \leftarrow S(\mathcal{B}_c, \mathcal{B}_{c'})$ for $c' < c$.
  Select the top-k past key blocks $\mathcal{T}_c$; form $\Omega_c \leftarrow \mathcal{T}_c \cup \{\text{own block } c\}$.
  Compute exact block scores $Z_c \leftarrow Q[\mathcal{B}_c]\, K[\Omega_c]^\top / \sqrt{d_h}$; apply a lower-triangular mask to the diagonal (own) block so each query attends only to non-future positions.
  Compute weights and denominators $A_c \leftarrow \exp(Z_c)$,
  $d_S[\mathcal{B}_c] \leftarrow A_c \mathbf{1}$.
  Compute normalized output $O_S[\mathcal{B}_c] \leftarrow \mathrm{diag}(d_S[\mathcal{B}_c] + \varepsilon)^{-1} A_c\, V[\Omega_c]$.
return O_S, d_S.

B.1 What we inherit from FlashAttention

FlashAttention computes exact attention without ever materializing the $N \times N$ score matrix $S = QK^\top/\sqrt{d_h}$ or the softmax weights $P = \mathrm{softmax}(S)$ in HBM. It tiles Q, K, V into blocks, and for each query tile streams over key tiles while maintaining, in on-chip SRAM, a running max m, a running denominator ℓ, and a running unnormalized output accumulator O; each new key tile rescales the accumulator by $\exp(m_{\mathrm{old}} - m_{\mathrm{new}})$ before adding its contribution. This *online softmax* recurrence produces the exact normalized output in one pass, reduces HBM traffic from $O(N^2)$ to $O(N^2/\sqrt{M})$ for SRAM of size M, and – crucially for training – stores only the length-N log-sum-exp (LSE)

statistics, not P, so the backward pass recomputes the attention weights tile-by-tile instead of reading them from HBM. The MASaLA block-sparse branch uses this recurrence unchanged in form: the same running max/denominator/accumulator state, the same rescaling, the same LSE output.

B.2 Departure 1: the block-sparse key loop

The only change to the FlashAttention recurrence in the sparse branch is the set of key tiles it streams over. A dense FlashAttention query tile iterates over all $\lceil N/B \rceil$ key tiles (or all not-masked-out tiles under a causal mask); the MASaLA sparse branch iterates only over the tiles in that query block's selected support Ω_c – the top-k scored blocks plus the forced own block (and, in the non-causal vision setting, the forced CLS block). Because $|\Omega_c|$ is a fixed budget independent of N, the branch performs $O(Ns)$ work and $O(Ns)$ HBM reads of K, V rather than $O(N^2)$. The selection itself (the block scorer and top-k) is computed before the attention kernel and passed in as a per-query list of key-block indices; it is the same selection described in Section 3.2. With the raw-key-space scorer of Section 3.2, this selection costs $\Theta(N \cdot \frac{N}{B} \cdot d_h)$ – quadratic in N, though reduced by the block size B – plus $O((N/B)^2)$ for top-k over N/B blocks; Appendix E gives the full accounting and its size relative to the branch cost. The branch emits the sparse output O_S and its per-token log-denominator $\log d_S$ (the LSE over the selected support), exactly the two quantities the fusion needs.

B.3 Departure 2: fusing the low-rank branch and the mass-aware epilogue

This is the main departure from a FlashAttention-style kernel, and the part that is specific to a sparse-plus-low-rank operator. A naive implementation would run three separate passes – the sparse branch, the low-rank branch, and a fusion epilogue that reads both outputs back from HBM to form $O = g_S m_S O_S + g_R O_R$ – materializing O_S and O_R (each $N \times d_h$) in HBM and paying three kernel launches. We instead fuse the low-rank branch and the fusion epilogue into the sparse kernel's epilogue, so that O_R is produced and consumed in registers and never reaches HBM.

The fusion is possible because the low-rank branch is associative in a way the sparse branch is not. RACE's output for query i is

$$O_{R,i} \;=\; \frac{\phi(q_i)^\top \big(\sum_\tau \phi(k_\tau) \otimes v_\tau\big)}{\phi(q_i)^\top \big(\sum_\tau \phi(k_\tau)\big)}, \tag{11}$$

where $\phi(\cdot) \in \mathbb{R}^{S_R}$ is the soft-hash feature map of Algorithm 2. The two key/value sums, $U = \sum_\tau \phi(k_\tau) \otimes v_\tau \in \mathbb{R}^{S_R \times d_h}$ and $z = \sum_\tau \phi(k_\tau) \in \mathbb{R}^{S_R}$, do not depend on i: they are computed once (in the causal case as the exclusive per-block prefix states of Algorithm 4) and kept resident, after which each query needs only an $O(S_R d_h)$ contraction against them. In the fused epilogue, for each query tile we (i) read the resident U and z, (ii) form O_R and its denominator d_R for that tile in registers via Equation (11), (iii) combine them with the sparse tile's O_S and d_S – already in registers from the recurrence – using the mass-aware weight

$$m_{S,i} \;=\; \sigma\big(\log d_{S,i} - \log d_{R,i} - \log \lambda_i\big), \tag{12}$$

computed in log space for numerical stability (equivalent to Equation (5) at $\varepsilon = 0$), and (iv) write out only the fused tile $O_i = g_{S,i} m_{S,i} O_{S,i} + g_{R,i} O_{R,i}$ together with the two scalars-per-token $\log d_{S,i}$ and $\log d_{R,i}$. Neither O_S nor O_R is written to HBM.

B.4 Backward pass without materializing branch outputs

The forward pass stores, per token, only the fused output O_i, the two log denominators, and the selection indices – not O_S, O_R, P_S, or the RACE feature matrices. The backward pass must therefore reconstruct whatever it needs. Two observations keep this linear in memory.

First, the gradient of the fusion with respect to a branch output is diagonal in the token index: from $O_i = g_{S,i} m_{S,i} O_{S,i} + g_{R,i} O_{R,i}$,

$$\frac{\partial \mathcal{L}}{\partial O_{S,i}} = g_{S,i} m_{S,i}\, \bar{O}_i, \qquad \frac{\partial \mathcal{L}}{\partial O_{R,i}} = g_{R,i}\, \bar{O}_i, \tag{13}$$

where $\bar{O}_i = \partial \mathcal{L} / \partial O_i$ is the incoming gradient. The denominators $d_{S,i}, d_{R,i}$ are detached when forming $m_{S,i}$ (so the branch outputs enter the backward pass only through the row-wise products in

Equation (13), not as full matrices, preserving the linear-memory argument above); λ_i is *not* detached, since it is learned (Section 3.3), so it receives the additional scalar gradient

$$\frac{\partial \mathcal{L}}{\partial \lambda_i} = g_{S,i}\langle \bar{O}_i, O_{S,i}\rangle \frac{\partial m_{S,i}}{\partial \lambda_i}, \qquad \frac{\partial m_{S,i}}{\partial \lambda_i} = -\frac{m_{S,i}(1-m_{S,i})}{\lambda_i} \quad (\text{at } \varepsilon = 0, \text{ Equation (12)}), \tag{14}$$

computed from the already-available $m_{S,i}, O_{S,i}, O_{R,i}$ and $\bar{O}_i$, so it adds no further branch materialization.

Second, each branch's own backward is itself a FlashAttention-style recompute. The sparse branch reruns its forward recurrence over the (stored) selected support to regenerate P_S tile-by-tile from the stored LSE, exactly as FlashAttention's backward does, and never reads a stored P_S. The low-rank branch recomputes U and z from $\phi(Q), \phi(K), V$ (themselves recomputed from Q, K) and backpropagates through Equation (11) with the $O(S_R d_h)$ contraction. Both recomputes produce, per token, length-d_h gradient rows that are accumulated in place; the largest transient is the RACE feature tensor of width $S_R = L_s R$, giving the constant-factor additive overhead over FlashAttention's own backward. No $N \times N$ or $N \times d_h$ branch intermediate is materialized in either direction, so total activation memory is $O(N(d_h + S_R))$, i.e. $O(N)$ for fixed d_h, L_s, γ – matching FlashAttention's asymptotic memory with an additive $O(NS_R)$ term.

B.5 Precision: why the kernel runs in bf16, not fp16

The kernel computes attention in reduced precision and casts the output back to fp32, as FlashAttention does. However, the low-rank branch imposes a constraint FlashAttention does not face. The soft-hash features $\phi(\cdot)$ are normalized probabilities over R buckets, so individual feature entries can be very small, and the gradients that flow back through the associative sums of Equation (11) into the Q, K projections can be correspondingly small. In fp16 (normal range down to $\sim 6 \times 10^{-5}$), with no gradient scaling in this pipeline, these gradients underflow to exactly zero – the branch's projections then receive no gradient and never train, silently. bf16 has the same exponent range as fp32 (only fewer mantissa bits), so it does not underflow; the kernel therefore uses bf16 by default. This is a genuine difference from a plain FlashAttention kernel, where the attention weights are $O(1)$-scale softmax probabilities and fp16 is typically adequate.

B.6 Memory accounting

MASaLA and FlashAttention reach out-of-memory at the same sequence length, so their memory scaling is identical, $O(N)$; the two differ only by a constant factor, which we account for here from first principles. We use the benchmarked configuration (B=1, H=4, N=65,536, d=384, padded head dimension 128, RACE L_s=4, γ=4 so S_R=64, gate hidden width 128, fp32 residual stream, bf16 kernel dtype, 4 layers).

What the fused kernel already avoids. The constant factor would be far larger under a naive two-branch implementation. A reference sparse-plus-low-rank kernel (including Scatterbrain's own pseudocode) materializes both branch outputs O_S and O_R before combining them, adding two $[B, H, N, d_h]$ tensors to the persistent set and two more ($\partial O_S, \partial O_R$) to the backward scratch. The MASaLA kernel stores none of these: the sparse branch never forms the $N \times N$ matrix (it loops only over selected blocks), the RACE branch and fusion epilogue run in one kernel so O_R is never written to HBM, and the backward recomputes O_S, O_R and immediately reduces them to the length-N row-wise inner products $\langle \bar{O}, O_S\rangle, \langle \bar{O}, O_R\rangle$ of Equation (13), so neither branch output nor its gradient is ever materialized in the forward or the backward.

The remaining constant factor. What remains is genuinely MASaLA-specific persistent state, retained from a layer's forward until its backward: the RACE signature tensors $\phi(Q), \phi(K)$ and the intermediates of their construction, the gate network's hidden activations, and the extra value-projection tensor the low-rank branch consumes. Summed over the four layers at the configuration above, this is roughly twice FlashAttention's per-layer persistent activation memory, plus a fp32 backward scratch (the gradient-accumulation buffers held one layer at a time) that sets the single-operator peak. Two properties of this overhead matter. First, it is a fixed set of tensors whose size does not grow with N faster than linearly, so it does not change the $O(N)$ scaling. Second, most of it is removable by implementation changes that do not alter the method: computing the RACE signatures

from the bf16 queries and keys the kernel already saves (rather than from separate fp32 copies), and recomputing the signatures in the backward, together bring the full-model memory footprint to within roughly 10% of FlashAttention in our analytic accounting; the residual single-operator gap is the fp32 backward scratch, whose removal is a numerics-sensitive kernel change we leave to future work.

Comparison caveat. All memory comparisons above pad the head dimension from 96 to 128 for both MASaLA and the matched FlashAttention baseline, so the ratio is a fair kernel-to-kernel comparison. Against a production FlashAttention running at the native head dimension 96, MASaLA's query, key, value, and output tensors are each about 33% wider; the reported ratios are therefore conservative in MASaLA's favor by that factor.

Measured model-level memory. We validate the accounting above by measuring peak training memory of a full 4-layer model (attention plus MLP blocks, batch size 1, bf16) as sequence length grows, comparing MASaLA against a matched FlashAttention baseline under an identical harness. Peak memory is `torch.cuda.max_memory_allocated` over a full forward–backward step. Figure 1 and Table 5 report the result. Three findings hold across the entire range, an $8 \times 16{,}384$-to-$262{,}144$ span:

1. **Identical scaling.** Both methods reach out-of-memory at the same sequence length ($524{,}288$) and both train successfully at $262{,}144$, confirming empirically that MASaLA inherits FlashAttention's $O(N)$ memory scaling rather than merely approaching it asymptotically.
2. **A flat, small constant factor.** MASaLA's peak training memory is $1.08\times$ FlashAttention's (a ratio of 0.925 in the FlashAttention/MASaLA convention), and this factor is essentially constant across the whole length range (0.925–0.927), rather than degrading as N grows. With the gate network's activations also recomputed in the backward, the MASaLA (RECOMPUTE ϕ) variant reaches $\approx 1.00\times$ (measured 1.00–1.01), i.e. parity with FlashAttention to within measurement noise.
3. **Where the overhead sits.** Splitting each measured peak into the forward-resident portion (a forward-only pass) and the additional memory that appears once the backward runs, MASaLA's overhead is roughly 17% forward-resident and 83% backward-side, stably across lengths. Both segments in Figure 1 are measured, not modeled; their sum is the measured total by construction.

We report the ratio honestly as an overhead: MASaLA uses modestly *more* peak memory than FlashAttention ($\approx 8\%$), reaching parity only when the signature and gate activations are both recomputed. This is the expected price of the second (low-rank) branch, and it does not change the memory scaling.

Table 5: Measured peak training memory (MB), 4-layer model, batch size 1, bf16. "fwd-resident" is the forward-only peak; "bwd-added" is the additional peak once the backward runs; their sum is the full-step peak. Ratio is FlashAttention / MASaLA of the full-step peak.

	FlashAttention		MASaLA		
N	total	(fwd/bwd)	total	(fwd/bwd)	ratio
16,384	1,356.6	411/946	1,463.5	427/1,036	0.927
32,768	2,621.1	729/1,892	2,832.6	767/2,066	0.925
65,536	5,154.6	1,371/3,784	5,568.4	1,441/4,127	0.926
131,072	10,217.6	2,649/7,569	11,044.5	2,795/8,250	0.925
262,144	20,346.6	5,211/15,136	21,993.3	5,497/16,496	0.925

B.7 WALL-CLOCK AND PEAK-MEMORY MEASUREMENTS

Figure 2 reports the measured forward- and backward-pass speedup of MASaLA over exact FlashAttention as a function of sequence length, at the attention-operator level (a single attention module, batch size 1, bf16, head dimension 96 padded to 128 for the fused kernel; the matched FlashAttention baseline pads identically, so the comparison is kernel-to-kernel). MASaLA is *slower* than exact FlashAttention at short lengths, where the quadratic cost FlashAttention pays is still small and MASaLA's two branches plus block scorer carry a fixed overhead: below the crossover length near

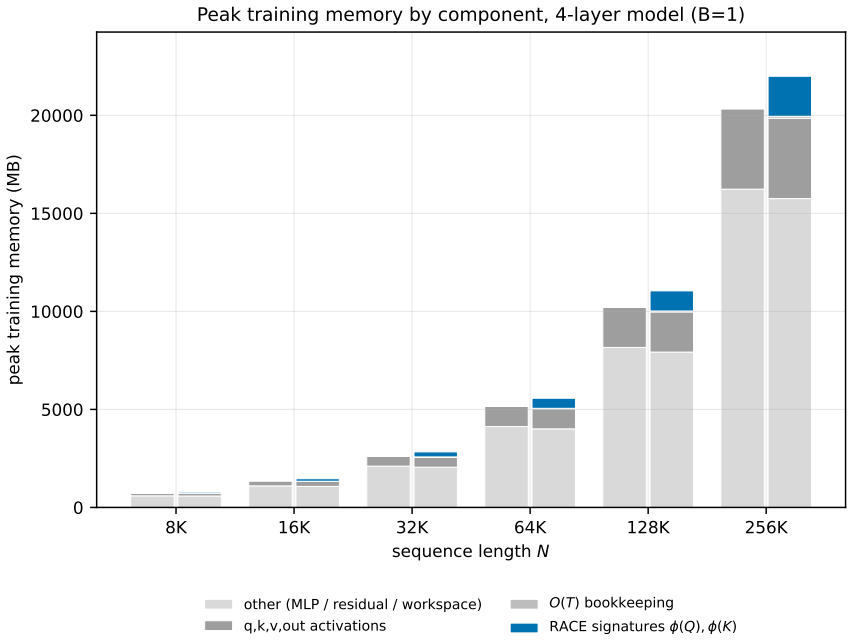


Figure 1: Measured peak training memory of a 4-layer model (batch size 1, bf16) versus sequence length. At each length, the *left* bar is FlashAttention and the *right* bar is MASaLA. Each bar is split into two directly-measured segments: memory resident during a forward-only pass (grey) and the additional memory that appears once the backward pass runs (blue). The ratio of total peak memory (FlashAttention / MASaLA) is essentially constant across the whole range at 0.93 (Table 5 gives the exact value at each length), so we do not repeat it at every pair of bars. Both methods reach out-of-memory at the same length (524,288), confirming identical $O(N)$ scaling.

$N \approx 10{,}600$ the ratio is under 1. Beyond the crossover the picture inverts and widens rapidly, because exact attention's cost grows as N^2 while MASaLA's grows linearly in the two branches (Appendix E): forward speedup reaches roughly $3.5\times$ at 32K and $13\times$ at 128K, with backward speedup tracking it a little lower ($3\times$ at 32K, $11\times$ at 128K). The two regimes are consistent with the method's intent – MASaLA is designed for the long-context lengths where exact attention is expensive, and it does not claim to beat a cheap exact kernel at short lengths, only to degrade gracefully there.

C PROOFS

Setup. Fix one attention head and let $Q, K, V \in \mathbb{R}^{N \times d_h}$ denote the *already-projected* queries and keys (i.e. $Q = XW_Q$, $K = XW_K$ for the layer's input X). For query i, let $\Omega_i \subseteq [N]$ be its block-sparse support (the blocks selected by the block scorer together with the forced own block), so that $i \in \Omega_i$ for every i; writing $\Omega = \{(i,j) : j \in \Omega_i\}$, this forcing means

$$\{(i,i) : i \in [N]\} \subseteq \Omega. \tag{15}$$

Write $\hat{A}^S$ for the row-normalized block-sparse attention matrix (supported on Ω) and $\hat{A}^R$ for the row-normalized RACE attention matrix of Algorithm 2. Since each RACE hash table contributes a rank-2^γ term, $\hat{A}^R$ satisfies

$$\operatorname{rank}(\hat{A}^R) \le r, \qquad r := L_s 2^\gamma. \tag{16}$$

The gated MASaLA fusion rule (Equation (7)) computes, row by row, $O_i = g_{S,i} m_{S,i} (\hat{A}^S V)_i + g_{R,i} (\hat{A}^R V)_i$, which is exactly $O = \hat{A} V$ for the combined operator

$$\hat{A} = \underbrace{G_S \hat{A}^S}_{=:B} + \underbrace{G_R \hat{A}^R}_{=:R}, \qquad \operatorname{supp}(B) \subseteq \Omega, \quad \operatorname{rank}(R) \le r, \tag{17}$$

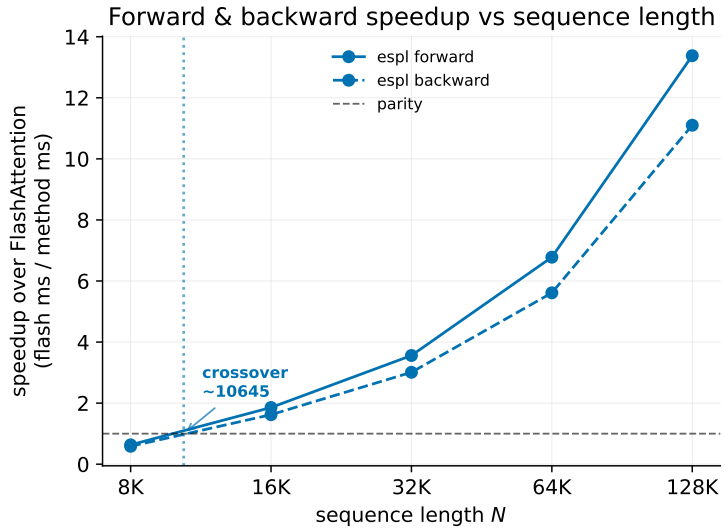


Figure 2: Forward and backward wall-clock speedup of MASaLA relative to exact FlashAttention versus sequence length N (attention-operator level, batch size 1, bf16; head dimension 96 padded to 128 for both the fused kernel and the matched FlashAttention baseline). Speedup is defined as (FlashAttention ms)/(MASaLA ms), so values above 1 mean MASaLA is faster; the solid line is the forward pass and the dashed line the backward pass. The horizontal dashed line marks parity ($1\times$) and the vertical dotted line marks the measured crossover length ($N \approx 10{,}600$), beyond which MASaLA is faster in both passes. The speedup grows with N because the exact baseline's cost is quadratic while MASaLA's is linear (Appendix E).

with $G_S = \mathrm{diag}(g_S \odot m_S)$, $G_R = \mathrm{diag}(g_R)$: the implemented MASaLA operator is literally a sparse-plus-low-rank matrix, with support fixed by the block scorer and forced own block, and rank budget fixed by the number of RACE tables and hash bits. The *ungated* variant of Section 3.3 (also evaluated in Table 2) drops the gates, giving

$$\hat{A}_{\mathrm{ung}} \;=\; M_S \hat{A}^S \;+\; \hat{A}^R, \qquad M_S = \mathrm{diag}(m_S). \tag{18}$$

We prove the rank-capacity guarantee (Lemma 1) for $\hat{A}_{\mathrm{ung}}$, where the argument is exact; a remark below addresses the gated operator $\hat{A}$. The absolute-continuity hypothesis of Lemma 1 holds whenever $Q = XW_Q, K = XW_K$ for X absolutely continuous and $[W_Q\ W_K]$ has full column rank, in particular whenever $2d_h \le d$.

This appendix gives the full statement and proof behind Lemma 1. It is convenient to first record the exact dense-attention matrix and three auxiliary facts that build up to the lemma, then two elementary tools used repeatedly, before giving the proof itself.

Dense attention. Let $S = QK^\top/\sqrt{d_h}$, $E = \exp(S)$ (entrywise), $D = \mathrm{diag}(E\mathbf{1})$, and let

$$A = D^{-1}E, \qquad A_{ij} = \frac{\exp(q_i^\top k_j/\sqrt{d_h})}{\sum_{\ell=1}^{N} \exp(q_i^\top k_\ell/\sqrt{d_h})} \tag{19}$$

be the exact dense-attention matrix, so that exact attention computes $O = AV$.

Auxiliary facts. Associate with Ω the bipartite graph G_Ω with left vertices (queries) and right vertices (keys) connected by the pairs in Ω, and let $\nu(\Omega)$ denote the size of a maximum matching in G_Ω. Let $k_\star = \min\{N, \nu(\Omega) + r\}$, with r as in Equation (16), and $\mathcal{H}_{\Omega,r} := \{B + R : \mathrm{supp}(B) \subseteq \Omega,\ \mathrm{rank}(R) \le r\}$ the sparse-plus-low-rank matrix class that $\hat{A}_{\mathrm{ung}}$ (Equation (18)) belongs to. We will establish, in order:

(a) **(Deterministic capacity.)** For every $B + R \in \mathcal{H}_{\Omega,r}$, $\mathrm{rank}(B + R) \le k_\star$, and this bound is attained by some $B + R \in \mathcal{H}_{\Omega,r}$.

(b) **(Generic attainment under free parameters.)** If the free entries of B on Ω and the factors $U, V \in \mathbb{R}^{N\times r}$ of $R = UV^\top$ are drawn from a distribution jointly absolutely continuous with respect to Lebesgue measure, then $\mathrm{rank}(B + R) = k_\star$ almost surely.

(c) **(Consequence of forced own-block support.)** Because Ω contains the full diagonal (Equation (15)), the identity matching gives $\nu(\Omega) = N$, hence $k_\star = N$ *regardless of* r: the block-sparse plus low-rank capacity equals the full rank of exact dense attention.

Lemma 1 itself is then the natural fourth step: it removes the free-parameter assumption of (b) and shows the same conclusion, $\mathrm{rank} = N$ almost surely, holds for the actual, data-dependent operator $\hat{A}(Q, K)$ — not just for B, R treated as independent free parameters.

Why a pure low-rank branch cannot reach this capacity. The contrast is sharpest against a pure low-rank branch. If (Q, K) is absolutely continuous, the same analytic-witness argument used to prove Lemma 1 below (with a strictly positive-definite Gram-kernel witness) shows $\mathrm{rank}(A) = N$ almost surely for the *exact* dense attention matrix A as well. By the Eckart–Young–Mirsky theorem (Eckart & Young, 1936; Mirsky, 1960), for every matrix $\tilde{A}$ with $\mathrm{rank}(\tilde{A}) \le r < N$,

$$\|A - \tilde{A}\|_2 \ge \sigma_{r+1}(A) > 0 \qquad \text{almost surely,} \tag{20}$$

so a rank-r branch such as a modified-RACE branch used alone *cannot* represent generic dense attention exactly, no matter how it is trained. Lemma 1 shows that forcing the query's own block into the block-sparse support is not merely a heuristic for retrieval recall: it is what supplies the perfect matching that removes this obstruction, giving the hybrid operator the same rank capacity as dense attention while the low-rank branch alone remains rank-bounded. We stress that full rank is necessary only for *exact* representation, and is neither necessary nor sufficient for small approximation error (Eckart–Young–Mirsky bounds the best rank-r error by $\sigma_{r+1}(A)$, which can be tiny): fact (c) above and Lemma 1 guarantee that the algebraic ceiling imposed by a rank-limited branch is removed, but say nothing by themselves about conditioning or approximation error, which is governed separately by how well the block scorer and RACE sketch track the true attention mass.

We now record two elementary facts used repeatedly in the proof.

Lemma 2 (Structural rank equals maximum matching)**.** *Let $\Omega \subseteq [N] \times [N]$ and let X be any matrix with $\mathrm{supp}(X) \subseteq \Omega$. Then $\mathrm{rank}(X) \le \nu(\Omega)$. Conversely, there exists a $0/1$ matrix X with $\mathrm{supp}(X) \subseteq \Omega$ and $\mathrm{rank}(X) = \nu(\Omega)$.*

Proof. If $\mathrm{rank}(X) = k$, some $k \times k$ minor of X is nonzero. Expanding this minor by the Leibniz formula, at least one of the $k!$ permutation terms is nonzero, i.e., there is a permutation π of a k-subset of columns matched to a k-subset of rows such that $X_{i,\pi(i)} \neq 0$ for all k indices i in the minor's rows. Since $\mathrm{supp}(X) \subseteq \Omega$, each such pair $(i, \pi(i)) \in \Omega$, and since π is a bijection between the two index sets these pairs form a matching of size k in G_Ω. Hence $k \le \nu(\Omega)$.

For the converse, take a maximum matching $M = \{(i_1, j_1), \ldots, (i_\nu, j_\nu)\}$ of G_Ω with $\nu = \nu(\Omega)$, and let X have $X_{i_\ell, j_\ell} = 1$ for $\ell = 1, \ldots, \nu$ and zero elsewhere. Since the i_ℓ are distinct and the j_ℓ are distinct, X restricted to rows $\{i_\ell\}$ and columns $\{j_\ell\}$ is a permutation matrix, so this $\nu \times \nu$ submatrix has determinant $\pm 1 \neq 0$, giving $\mathrm{rank}(X) \ge \nu$. Combined with the first half, $\mathrm{rank}(X) = \nu(\Omega)$. □

Lemma 3 (Zero sets of nonzero polynomials and real-analytic functions)**.** *If $p : \mathbb{R}^m \to \mathbb{R}$ is a polynomial that is not identically zero, then $\{x \in \mathbb{R}^m : p(x) = 0\}$ has Lebesgue measure zero. More generally, the same conclusion holds if p is replaced by any real-analytic function $f : \mathbb{R}^m \to \mathbb{R}$ that is not identically zero (Mityagin, 2020).*

Proof sketch. For polynomials, induct on m. The case $m = 1$ is the fact that a nonzero univariate polynomial of degree d has at most d roots, a finite (measure-zero) set. For $m > 1$, write p as a polynomial in x_m with coefficients that are polynomials in $x_1, \ldots, x_{m-1}$; since $p \not\equiv 0$, some coefficient polynomial q is not identically zero, and by the inductive hypothesis $\{q = 0\} \subset \mathbb{R}^{m-1}$ has measure zero. For every $(x_1, \ldots, x_{m-1})$ with $q \neq 0$, the slice $\{x_m : p(x) = 0\}$ is finite. By Fubini's theorem, the total measure of $\{p = 0\}$ is the integral over $\mathbb{R}^{m-1}$ of the (a.e. zero or finite, hence measure-zero) slice measure, which is zero. The real-analytic case follows from the identity theorem for real-analytic functions: if $f \not\equiv 0$ vanished on a set of positive measure, it would vanish

on an open ball (by Lebesgue density), and analytic continuation would force $f \equiv 0$ on its connected domain, a contradiction. □

Proof of facts (a)–(c) and Lemma 1. We first prove auxiliary facts (a)–(c) in order, then use them to prove the lemma itself.

Part (a), upper bound. Fix $B + R \in \mathcal{H}_{\Omega,r}$. By subadditivity of matrix rank under addition,

$$\operatorname{rank}(B + R) \le \operatorname{rank}(B) + \operatorname{rank}(R). \tag{21}$$

Since $\operatorname{supp}(B) \subseteq \Omega$, Lemma 2 gives $\operatorname{rank}(B) \le \nu(\Omega)$. By hypothesis $\operatorname{rank}(R) \le r$. Hence $\operatorname{rank}(B+R) \le \nu(\Omega)+r$. Since also $\operatorname{rank}(B+R) \le N$ trivially (it is an $N \times N$ matrix), we obtain $\operatorname{rank}(B + R) \le \min\{N, \nu(\Omega) + r\} = k_\star$.

Part (a), attainability. By Lemma 2, choose a $0/1$ matrix B_0 supported on a maximum matching M of G_Ω, so $\operatorname{rank}(B_0) = \nu(\Omega)$ and B_0 restricted to the $\nu(\Omega)$ matched rows and columns is a permutation matrix. The remaining $N - \nu(\Omega)$ rows and $N - \nu(\Omega)$ columns are unmatched by M. Let $s = \min\{r,\, N - \nu(\Omega)\}$, pick s of the unmatched rows and s of the unmatched columns, and let R_0 be the $N \times N$ matrix that is a permutation matrix on exactly this disjoint $s \times s$ block and zero elsewhere; by construction $\operatorname{rank}(R_0) = s \le r$ (pad with zero rows/columns of the low-rank factors if $s < r$, which does not violate $\operatorname{rank}(R_0) \le r$). Because B_0 and R_0 occupy disjoint rows and disjoint columns (the matched block for B_0, the newly paired block for R_0, and the rest zero), $B_0 + R_0$ is itself a partial permutation matrix of rank $\nu(\Omega) + s = k_\star$. Hence $k_\star$ is attained in $\mathcal{H}_{\Omega,r}$, completing part (a).

Part (b). Consider the $k_\star \times k_\star$ minor of $B + UV^\top$ formed by the rows and columns used in the attainability construction above (the matched rows/columns of B_0 together with the s extra rows/columns of $R_0 = U_0 V_0^\top$ for some fixed U_0, V_0). As a function of the free entries of B on Ω and the entries of U, V, this minor is a polynomial p. By the attainability construction, p evaluated at (B_0, U_0, V_0) equals $\pm 1 \neq 0$, so p is not the zero polynomial. By Lemma 3, $\{p = 0\}$ has Lebesgue measure zero, so under a jointly absolutely continuous law on the free parameters, $p \neq 0$ almost surely, giving $\operatorname{rank}(B + UV^\top) \ge k_\star$ almost surely. Combined with the deterministic upper bound of part (a), $\operatorname{rank}(B + UV^\top) = k_\star$ almost surely.

Part (c). By construction (Equation (15)), $(i, i) \in \Omega$ for every $i \in [N]$, so the identity permutation $\{(i, i) : i \in [N]\}$ is itself a matching of G_Ω of size N. Since no matching can exceed size N, $\nu(\Omega) = N$. Substituting into $k_\star = \min\{N, \nu(\Omega) + r\} = \min\{N, N + r\} = N$, which holds for every $r \ge 0$.

Proof of the lemma. We now remove the free-parameter assumption of fact (b) and work directly with the coupled, data-dependent operator $\hat{A}_{\mathrm{ung}}(Q, K)$ of Equation (18).

Step 1 (analyticity for a frozen support pattern). The support Ω is chosen by a top-k rule and so is only piecewise constant in (Q, K); fix, for this step, one of the finitely many admissible selection patterns P (a pattern assigns, to each query block, its own block plus k selected key blocks; every such P contains the full diagonal by the own-block-forcing rule, Equation (15)), and let $\hat{A}_{\mathrm{ung},P}$ denote $\hat{A}_{\mathrm{ung}}$ with Ω frozen to P. Every coordinate map from (Q, K) to an entry of $\hat{A}_{\mathrm{ung},P}(Q, K)$ factors through: (i) the score map $(q_i, k_j) \mapsto \exp(q_i^\top k_j/\sqrt{d_h})$, a composition of a bilinear map and $\exp$, both analytic; (ii) the block-sparse row normalization $\hat{A}^S_{ij} = E_{ij}\mathbf{1}[(i, j) \in P_i]/\sum_{\ell \in P_i} E_{i\ell}$, a ratio of analytic functions whose denominator is a finite sum of strictly positive exponentials, hence bounded away from zero on bounded sets and analytic everywhere; (iii) the RACE features $\Phi^{(\ell)}_Q, \Phi^{(\ell)}_K$ of Algorithm 2, each coordinate a ratio of exponentials of $\tanh(W^{(\ell)}x)$ with a denominator that is a sum of $R = 2^\gamma$ strictly positive terms, hence analytic; (iv) the RACE row normalization $\operatorname{diag}(\mathrm{Den} + \varepsilon)^{-1}$, analytic since $\mathrm{Den}_i + \varepsilon > 0$ always; and (v) the mass-aware weight $m_{S,i} = d_{S,i}/(d_{S,i} + \lambda_i d_{R,i} + \varepsilon)$, analytic in (Q, K) since $d_{S,i}, d_{R,i}, \lambda_i$ are each analytic and the denominator is bounded away from zero (for the input-dependent variant, $\lambda_i = c + \sigma(w^\top q_i + b)$ is a composition of an affine map and the analytic sigmoid). A finite composition and sum of real-analytic functions with nowhere-vanishing analytic denominators is real-analytic. Hence every entry of $\hat{A}_{\mathrm{ung},P}(Q, K)$, and therefore $\det \hat{A}_{\mathrm{ung},P}(Q, K)$, is real-analytic on $\mathbb{R}^{2Nd_h}$, for every admissible pattern P.

Step 2 (a common witness family for every pattern). Assume $d_h \geq 2$ and fix N pairwise *distinct* unit vectors $x_1, \ldots, x_N \in \mathbb{R}^{d_h}$ (these exist for any N once $d_h \geq 2$, unlike an orthonormal set, which would require $d_h \geq N$ and fails in our experiments, where $d_h \ll N$); let $\delta := \min_{i \neq j}(1 - x_i^\top x_j) > 0$. Set $Q = K$ with rows $q_i = k_i = \tau x_i$ for a scalar $\tau > 0$ to be sent to infinity. For $j \neq i$,

$$\frac{E_{ij}}{E_{ii}} = \exp\Big(- \frac{\tau^2(1 - x_i^\top x_j)}{\sqrt{d_h}} \Big) \leq \exp\Big(- \frac{\tau^2 \delta}{\sqrt{d_h}} \Big) \;\longrightarrow\; 0 \quad \text{as } \tau \to \infty, \tag{22}$$

uniformly over the at most $N - 1$ off-diagonal terms in any row. This bound is independent of which pattern P is frozen (it only uses $i \in P_i$, which every admissible P satisfies), so for *every* admissible P,

$$\hat{A}^S_{ii} = \Big(1 + \sum_{\ell \in P_i \setminus \{i\}} E_{i\ell}/E_{ii}\Big)^{-1} \longrightarrow 1, \qquad \hat{A}^S_{ij} \to 0 \; (j \neq i), \quad \text{as } \tau \to \infty. \tag{23}$$

Likewise $m_{S,i} \to 1$ for every i: $d_{S,i} \geq E_{ii} \to \infty$ while $d_{R,i}$ (a RACE denominator built from bounded softmax features) stays bounded, and λ_i stays in a fixed range independent of τ – if λ_i is fixed or scalar-learned it does not depend on (Q, K) at all, and if $\lambda_i = c + \sigma(w^\top q_i + b)$ is input-dependent, $\sigma(\cdot) \in (0, 1)$ bounds it in $(c, c + 1)$ for every τ, since the sigmoid saturates rather than diverging as its argument $\tau(w^\top x_i) \to \pm\infty$. Hence $M_S \hat{A}^S_P \to I$ entrywise as $\tau \to \infty$, for every admissible P, and

$$\hat{A}_{\mathrm{ung},P}(Q, K) \;\longrightarrow\; I + \hat{A}^R_\infty \qquad \text{entrywise as } \tau \to \infty, \tag{24}$$

where $\hat{A}^R_\infty$ is the bounded, entrywise-nonnegative limit of $\hat{A}^R$ along this path (identical for every P, since $\hat{A}^R$ does not depend on Ω). Since $\hat{A}^R$'s rows sum to $\mathrm{Den}_i/(\mathrm{Den}_i + \varepsilon) < 1$ (Algorithm 2), $\|\hat{A}^R_\infty\|_\infty < 1$: for any $x \neq 0$ with $(I + \hat{A}^R_\infty)x = 0$, $x = -\hat{A}^R_\infty x$ gives $\|x\|_\infty \leq \|\hat{A}^R_\infty\|_\infty \|x\|_\infty < \|x\|_\infty$, a contradiction, so $I + \hat{A}^R_\infty$ is invertible. By continuity of the determinant, for *every* admissible pattern P there is a finite τ_P large enough that $\det \hat{A}_{\mathrm{ung},P}(Q, K) \neq 0$ at $(Q, K) = (\tau_P x_i, \tau_P x_i)_{i=1}^N$.

Step 3 (conclusion via a finite union of null sets). Step 2 exhibits, for every admissible pattern P, a witness at which $\det \hat{A}_{\mathrm{ung},P} \neq 0$; so by Step 1, each real-analytic function $\det \hat{A}_{\mathrm{ung},P}(\cdot, \cdot)$ is not identically zero on $\mathbb{R}^{2Nd_h}$, and by Lemma 3 each set $\{\det \hat{A}_{\mathrm{ung},P} = 0\}$ has Lebesgue measure zero. Since $\hat{A}_{\mathrm{ung}}(Q, K) = \hat{A}_{\mathrm{ung},P(Q,K)}(Q, K)$ for whichever pattern $P(Q, K)$ the scorer realizes,

$$\{\det \hat{A}_{\mathrm{ung}} = 0\} \;\subseteq\; \bigcup_P \{\det \hat{A}_{\mathrm{ung},P} = 0\}, \tag{25}$$

a *finite* union (there are finitely many admissible patterns) of Lebesgue-null sets, hence itself null. So for (Q, K) absolutely continuous,

$$\Pr\big[\det \hat{A}_{\mathrm{ung}}(Q, K) = 0\big] = 0 \quad \Longleftrightarrow \quad \Pr\big[\operatorname{rank}(\hat{A}_{\mathrm{ung}}(Q, K)) = N\big] = 1, \tag{26}$$

which is Equation (8). This completes the proof of Lemma 1, and with it the derivation of facts (a)–(c) and the lemma in full. □

Remark on the gated operator. The gated operator $\hat{A}$ of Equation (17), which is what the experiments in Section 7 report alongside the ungated variant, replaces $M_S \hat{A}^S \to I$ above with $G_S M_S \hat{A}^S \to \mathrm{diag}(g_S)$ and $\hat{A}^R \to G_R \hat{A}^R_\infty$ for the gates' limiting values along the same witness path. Because the gate logits $G_\theta(q_i)$ depend on $q_i = \tau x_i$ and can in principle grow with τ, $g_{S,i}$ and $g_{R,i}$ need not converge to fixed limits as $\tau \to \infty$ along this particular path, so the argument above does not directly transfer; establishing the same guarantee for $\hat{A}$ would require jointly perturbing (Q, K) and the gate parameters θ (e.g. via a Fubini-type argument over θ for each fixed large τ) and is left to future work. We are not aware of a mechanism by which gating would *destroy* the rank capacity established for $\hat{A}_{\mathrm{ung}}$, and Table 2 evaluates both variants empirically.

D Experiment Hyperparameters

We evaluate all attention variants using encoder-style, non-causal Transformer architectures. Unless otherwise stated, we train with cross-entropy loss and the AdamW optimizer. We use dropout 0.1, set

`qkv_bias=False`, and fix the random seed to 42. For the hybrid methods, the gate is a two-layer MLP with SiLU activation and two independent sigmoid outputs, one for the block-sparse branch and one for the RACE branch. We do not normalize the gates.

For the image experiments, Oxford-IIIT Pet and Flowers-102 use 512×512 RGB images with patch size 4, giving $128 \times 128 = 16{,}384$ image tokens. For Food-101, we use the long-image setting with 512×512 inputs, patch size 4, and $N = 16{,}384$. For LRA-Image, we use CIFAR-10 as a 32×32 grayscale pixel sequence with patch size 1, giving $N = 1024$. The vision models use a learnable class token and learnable positional embeddings.

For ArXiv classification, we tokenize documents using a basic English tokenizer, keep documents with at least 1,000 raw tokens, and perform class-balanced streaming packing. Documents from the same class are concatenated until the target sequence length is reached. For the reported ArXiv classification setting, we use $N = 32{,}000$, a vocabulary limit of 50,000, and packed train/test examples produced from the class-balanced long-document subset.

For MASaLA, the sparse branch is the block-sparse module of Section 3.2 and the low-rank branch is the modified RACE branch of Section 3.1. The output is computed as $O = g_S\, m_S\, O_S + g_R\, O_R$ (Equation (7)), where g_S and g_R are token-wise sigmoid gates and $m_S = d_S/(d_S + \lambda d_R + \epsilon)$. The reported results use the input-dependent $\lambda_i = c + \sigma(w^\top q_i + b)$, where c is initialized to 0.3, is learnable, and is constrained to be nonnegative; the bias is initialized to 0, so that with $c = 0.3$ the initial value of $\sigma(w^\top q_i + b) \approx 0.5$ gives $\lambda_i \approx 0.8$ at initialization; we detach q_i from the λ path, and initialize w with standard deviation 10^{-3}. A scalar variant, $\lambda = \exp(\ell_\lambda)$ initialized at $\lambda = 1.0$ and learned jointly with the rest of the model, is also implemented but was not used for the reported results.

For IMDB, we build a long-review subset at $N = 512$. Reviews with length between N and $2N$ are kept directly, longer reviews are split into overlapping windows with stride $N/2$, and shorter reviews from the same class are concatenated until they reach the target length. During training, we apply light augmentation using random deletion with probability 0.05 or random token swaps.

For the vision datasets, we use standard data augmentation. Oxford-IIIT Pet, Flowers-102, and Food-101 use random resized cropping and horizontal flipping for training, and resize followed by center crop for validation. RGB image datasets are normalized with ImageNet mean and standard deviation.

Table 6: Main dataset-level hyperparameters. Here N denotes the input sequence length after tokenization or patchification.

Dataset / Task	N	Layers	Heads	d	MLP dim	Batch	Grad. accum.	LR	Epochs
IMDB	512	1	2	128	512	32	1	1×10^{-5}	150
LRA-Image (CIFAR-10)	1,024	2	4	384	1536	32	1–2	6×10^{-4}	150
Oxford-IIIT Pet	16,384	2	4	384	1536	32	1–2	6×10^{-4}	150
Flowers-102	16,384	2	4	384	1536	32	1–2	6×10^{-4}	150
Food-101	16,384	8	8	512	2048	8	4	3×10^{-4}	100
ArXiv classification	32,000	4	4	256	1024	8	16	3×10^{-4}	33

Table 7: Attention-specific hyperparameters.

Component	Text settings	Vision settings
RACE hash bits γ	4 for ArXiv, 3 for IMDB	2
RACE tables L_s	4 for ArXiv, 2 for IMDB	5
RACE ensembles M	1 for ArXiv, 2 for IMDB	1
Block-sparse block size B	256 for ArXiv, 32 for IMDB	32
Block-sparse min length	4096 for ArXiv, 256 for IMDB	256
Gate hidden dim	64	128
Gate normalization	False	False
ϵ for denominator correction	10^{-6}	10^{-6}

E Complexity Derivation and Numerical Example

This appendix derives Equation (10) branch by branch, then gives a concrete numerical example.

Per-branch cost. FULL attention computes all query–key interactions:

$$C_{\text{FULL}} = \Theta(N^2), \qquad M_{\text{FULL}} = \Theta(N^2). \tag{27}$$

In BLOCK-SPARSE, each query attends only to a fixed set of selected and forced blocks. Writing s for the per-query key budget (the number of selected plus forced blocks, times the block size B), each query attends to at most s keys:

$$C_{\text{BLOCK-SPARSE}} = \Theta(Ns), \qquad M_{\text{BLOCK-SPARSE}} = \Theta(Ns). \tag{28}$$

Because the budget s is fixed (rather than a fixed fraction of N), this branch cost is linear in N. For RACE, let L_s be the number of hash tables and γ be the number of hash bits per table. Each table has 2^γ buckets, hence the total number of bucket features is $S_R = L_s 2^\gamma$. RACE does not materialize an $N \times N$ matrix; it builds global bucket summaries and each query reads from these S_R features:

$$C_{\text{RACE}} = \Theta(NL_s 2^\gamma), \qquad M_{\text{RACE}} = \Theta(NL_s 2^\gamma). \tag{29}$$

MASaLA combines BLOCK-SPARSE and RACE:

$$C_{\text{MASALA}} = \Theta\big(Ns + NL_s 2^\gamma\big) = \Theta\big(N(s + L_s 2^\gamma)\big), \qquad M_{\text{MASALA}} = \Theta\big(N(s + L_s 2^\gamma)\big), \tag{30}$$

which is Equation (10) with $S_R = L_s 2^\gamma$. The ablation BS+RACE has the same asymptotic cost as MASaLA, but removes the mass-aware rescaling by setting $m_{\text{sparse}} = 1$; this changes neither branch's cost. For fixed s, L_s, and γ, the two branches themselves (BLOCK-SPARSE and RACE) are linear in N, while FULL attention is quadratic.

Attention interaction count. We give a concrete example to illustrate the scale of the block-sparse–low-rank construction's savings. For $N = 32{,}000$, $s = 256$, $L_s = 4$, $\gamma = 4$, full attention computes $N^2 = 1.024 \times 10^9$ query–key interactions per head. The block-sparse branch computes $Ns = 32{,}000 \cdot 256 = 8.192 \times 10^6$ selected interactions (0.8% of full attention); the RACE branch uses $NL_s 2^\gamma = 32{,}000 \cdot 4 \cdot 16 = 2.048 \times 10^6$ bucket interactions (0.2%); together, $N(s + L_s 2^\gamma) = 1.024 \times 10^7$ interactions, 1.0% of full attention.

Parameter overhead. The additional learnable parameters introduced by MASaLA are small compared with the base Transformer layer. The standard attention projections $W_Q, W_K, W_V, W_O \in \mathbb{R}^{d \times d}$ already contribute $O(d^2)$ parameters. In contrast, the scalar-λ version of MASaLA adds only $O(1)$ parameters, a query-dependent λ version adds $O(d)$ parameters, and the gate MLP adds $O(dg)$ parameters for gate hidden width $g \ll d$. Thus, the parameter overhead is negligible relative to the base attention projections, while the main savings come from reducing the input-dependent attention computation.