SMAT-ATTENTION: STRUCTURED LONG-CONTEXT SEQUENCE MODELING

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-context sequence models face a fundamental tradeoff: softmax attention uses flexible token-level interactions at quadratic cost, whereas linear attention obtains linear-time training and constant-time decoding by compressing history into a fixed-size state. In this work, we ask whether we can connect these regimes through a tunable notion of structure. To this end, we introduce Structured Matrix Attention (SMat-Attention) via a family of causal masks with structured long-range routing whose row supports have VC-dimension d. In our construction, $d = 1$ recovers the standard causal mask, and increasing d permits richer subset-routing patterns. We give chunkwise forward and backward algorithms to enable hardware-efficiency. For sequences of length T, the hard-routing construction takes $O(T^{2-3/d} + T)$ work, despite the mask being dense, for our prescribed family. In fixed-horizon streaming, decoding after the distant prefix takes constant time per token using $O(T^{1-1/d})$ cached states. SMat-Attention therefore makes VC-dimension an explicit knob governing access-pattern complexity, prefill cost, and decoding memory. Empirically, subset-routing and rule-assisted multi-key retrieval experiments illustrate the masks' routing expressiveness. Extensions to Mamba-2 and Gated DeltaNet using learned routing with top-k query reads retain subquadratic prefill, improve recall accuracy over the backbones in several settings, and achieve comparable small-scale language-modeling performance.

1 INTRODUCTION

Attention is a foundational building block of modern deep learning (Bahdanau et al., 2014) and serves as the core mechanism for modeling token interactions in Transformer architectures (Vaswani et al., 2017). Given key, query, and value matrices $\mathbf{Q}, \mathbf{K}$, and $\mathbf{V}$, softmax attention computes

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}. \tag{1}$$

This operation gives each query direct access to token-level information, but its prefill computation grows quadratically with sequence length and its decoding cache grows linearly (Vaswani et al., 2017). However, fundamentally, long-context sequence modeling requires retaining useful information and selecting which parts of the past should influence each query. Hardware-aware kernels improve execution efficiency (Dao, 2023; Shah et al., 2024; Liu et al., 2024; Kwon et al., 2023), while sparse methods such as Native Sparse Attention and MoBA reduce the interactions evaluated for each query (Yuan et al., 2025; Lu et al., 2025).

Recurrent alternatives such as linear attention address these costs by compressing the history into a fixed-size recurrent state (Katharopoulos et al., 2020). Modern variants improve how the model maintains this state. Structured state-space models (SSMs) (Fu et al., 2023; Gu et al., 2022) compress history with time-invariant recurrences; Mamba and Mamba-2 make this recurrence input-dependent through selective gating (Gu and Dao, 2023; Dao and Gu, 2024), while DeltaNet and Gated DeltaNet use structured transition matrices (Schlag et al., 2021; Yang et al., 2024a; 2025) that update via the delta rule (Schmidhuber, 1992; Widrow and Hoff, 1960). These mechanisms improve retention and retrieval; however, their fixed-size hidden state still constrains associative recall over long contexts (Arora et al., 2023).

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.

These advances highlight the role of structure in efficient sequence modeling. For instance, linear attention exploits its causal prefix structure to reuse accumulated key–value summaries, yielding $O(T)$ computation. Gated variants, in turn, extend this approach through semiseparable structure (Dao and Gu, 2024), while long-convolution models exploit Toeplitz structure to compute their outputs in $O(T \log T)$ time using FFT (Poli et al., 2023; Qin et al., 2023). Log-Linear Attention (Guo et al., 2025) further expands this design space by changing the organization of memory: it organizes recurrent summaries through a Fenwick-tree hierarchy, achieving $O(T \log T)$ computation and $O(\log T)$ decoding memory. Other recent approaches route tokens among multiple recurrent states (Du et al., 2026) or let the compressed memory grow with context length (Behrouz et al., 2026; Goldstein et al., 2026), or as a latent vector (Anand et al., 2026). These approaches motivate studying not only how much memory a model retains, but also which subsets of stored information each query can access. Therefore, we investigate the following question: *can the combinatorial richness of long-range access patterns be made an explicit architectural parameter, with corresponding guarantees on computation and memory?*

We study structured long-range access as an intermediate regime and how its complexity governs computation and memory. To make this complexity explicit, we use the VC dimension of a causal mask's row supports (Kearns and Vazirani, 1994; Vapnik and Chervonenkis, 1971). Each row specifies the keys available to a query; under this lens, the VC dimension measures the largest number of keys on which the rows realize every possible subset. Recent connections between VC dimension and matrix multiplication (Anand et al., 2025) motivate constructing attention mechanisms that couple this combinatorial parameter to computational guarantees.

We introduce Structured Matrix Attention (SMat-Attention), a family of causal attention masks built from point-hyperplane incidences over finite fields, whose row-support VC dimension d explicitly controls long-range access complexity. Our framework recovers ordinary causal masking when $d = 1$, and exploits additional structure to support richer access patterns with subquadratic attention. Building on structured masked-attention formulations (Choromanski et al., 2023), we establish the following results:

1. We exploit the resulting incidence structure to derive chunkwise forward and backward algorithms. For sequences of length T, we show that this mechanism takes $O(T^{2-3/d} + T)$ work, yielding $O(T)$-attention for $d \leq 3$ and $O(T^{2-3/d})$-attention for $d > 3$, despite the full causal mask having $\Theta(T^2)$ nonzero entries.
2. We show that after processing a fixed distant prefix, SMat-Attention supports T-independent per-token decoding using $O(T^{1-1/d})$ cached states, revealing an explicit tradeoff between long-range access complexity and memory.
3. We extend the construction to Mamba-2 and Gated DeltaNet with learned content hashing and a learned four-read selector. The extension inherits the tabulation, cache and VC bounds of the above; its selector adds $\Theta(T^{2-1/d})$ prefill work and $O(T^{1-1/d})$ per decoded token. Controlled tasks show benefits consistent with increased routing expressiveness; learned SMat extensions improve mean recall accuracy over native backbones in several tested settings and remain competitive on small-scale PG-19 language modeling.

2 PRELIMINARIES

Let T be the length of the input sequence. Following Vaswani et al. (2017), attention linearly projects the input tokens into $\mathbf{Q} \in \mathbb{R}^{T \times d_{QK}}$, $\mathbf{K} \in \mathbb{R}^{T \times d_{QK}}$ and $\mathbf{V} \in \mathbb{R}^{T \times d_v}$, the queries, keys and values. Following Choromanski et al. (2023), the *general masked kernel attention* is

$$\mathsf{Att}_K(\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{M}) = \mathbf{D}^{-1}\mathbf{A}\mathbf{V}, \qquad \mathbf{A} = \mathbf{M} \odot \mathcal{K}(\mathbf{Q}, \mathbf{K}), \qquad \mathbf{D} = \mathsf{diag}(\mathbf{A}\mathbf{1}_T),$$

where $\odot$ is the entrywise product, $\mathcal{K} : \mathbb{R}^{d_{QK}} \times \mathbb{R}^{d_{QK}} \to \mathbb{R}$ is a kernel, $\mathcal{K}(\mathbf{Q}, \mathbf{K})_{ij} = K(\mathbf{q}_i, \mathbf{k}_j)$ for $\mathbf{q}_i$ the ith row of $\mathbf{Q}$ and $\mathbf{k}_j$ the jth row of $\mathbf{K}$, and $\mathbf{1}_T$ is the all-ones vector. Softmax attention is the special case $K(x, y) = \exp(\frac{x^\top y}{\sqrt{d_{QK}}})$ where $\mathbf{M} = \exp(\mathbf{N})$ entrywise and $\mathbf{N}$ is the logits mask.

Finite-feature attention. Assuming the kernel has a nonnegative feature factorization of dimension r, $\mathcal{K}(q, k) = \phi_{\mathbf{Q}}(q)^\top \phi_{\mathbf{K}}(k)$, where $\phi_{\mathbf{Q}}(q), \phi_{\mathbf{K}}(k) \in \mathbb{R}^r_{\geq 0}$. Let $\phi_i := \phi_{\mathbf{Q}}(q_i)$ and $\psi_j := h_j \phi_{\mathbf{K}}(k_j)$,

where $h_j \geq 0$ is an optional key gate (set $h_j \equiv 1$ for ungated attention). The normalizer is carried along with the values by appending a constant coordinate: with $p := d_v + 1$,

$$\bar{v}_j := \begin{bmatrix} v_j \\ 1 \end{bmatrix} \in \mathbb{R}^p, \qquad \mathbf{Z}_j := \psi_j \bar{v}_j^\top \in \mathbb{R}^{r \times p}. \tag{2}$$

For any nonnegative mask $\mathbf{M}$ the *augmented output* and the attention output are

$$\bar{y}_i = \phi_i^\top \sum_{j=1}^{T} \mathbf{M}_{ij} \mathbf{Z}_j \in \mathbb{R}^p, \qquad o_i = \frac{\bar{y}_{i,1:d_v}}{\bar{y}_{i,p}}. \tag{3}$$

Through the augmentation, $\bar{y}_i$ is *linear* in $\mathbf{M}$: the last coordinate accumulates the denominator along with the numerator, and the single nonlinearity is the final division. So if $\mathbf{M} = \sum_\ell \mathbf{M}^{[\ell]}$, the contributions $\bar{y}_i^{[\ell]}$ can be computed independently, in different orders and with different computational kernels, as long as they are summed before the division. Section 3.1 does this with two summands. Importantly, we require the feature map to be finite and nonnegative, so softmax attention is covered only through a kernel approximation such as Choromanski et al. (2022).

VC dimension of a mask. A binary mask $\mathbf{M} \in \{0,1\}^{T \times T}$ defines a set system on the key indices: row i is the set $S_i(\mathbf{M}) = \{j \in [T] : \mathbf{M}_{ij} = 1\}$ of keys visible to query i. If we let $\mathcal{S}(\mathbf{M}) = \{S_1(\mathbf{M}), \ldots, S_T(\mathbf{M})\}$, then $\mathrm{VC}(\mathbf{M})$ is the VC dimension of $\mathcal{S}(\mathbf{M})$, i.e. the largest k for which some set of k keys is shattered by the rows. The causal mask $\mathbf{L}_T$ has $\mathrm{VC} = 1$: its rows are the prefixes $\{1, \ldots, i\}$, which are totally ordered, so no two keys can be shattered, i.e. no query sees a later key without also seeing every earlier one. On the other hand, an unconstrained T-row mask can have VC-dimension as large as $\lfloor \log T \rfloor$. The parameter d interpolates between these, and Theorems 3.2 and 3.3 price the interpolation.

3 SMAT-ATTENTION

We construct a family of causal masks $\mathbf{M}^{(1)}, \mathbf{M}^{(2)}, \ldots$ indexed by their VC dimension. We show that attention under $\mathbf{M}^{(d)}$ can be computed in $O(T^{2-3/d} + T)$ work. Each mask is built from incidences between points and hyperplanes over a finite field, which gives it a computationally favorable structure. For $d \leq 3$ the forward pass is linear in the sequence length, and decoding runs from a cache of $O(T^{1-1/d})$ states.

Block form. Let n be the number of *distant* tokens, and $m = T - n$ be the remaining *recent* tokens. Let $\mathbf{L}_s$ be the inclusive lower-triangular all-ones matrix of order s. Every mask in the family has the form

$$\mathbf{M}^{(d)} = \begin{pmatrix} \mathbf{L}_n & \mathbf{0} \\ \mathbf{G} & \mathbf{L}_m \end{pmatrix}, \qquad \mathbf{G} \in \{0,1\}^{m \times n}, \tag{4}$$

so distant and recent tokens are each causally masked, and $\mathbf{G}$ encodes all of the long-range interaction.

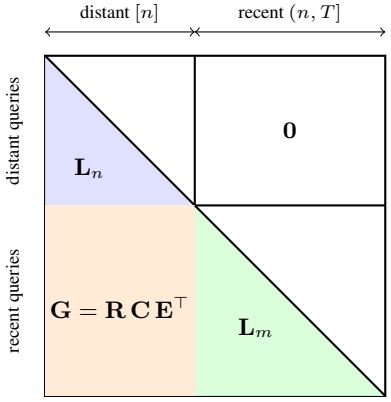

Figure 1: Block layout of $\mathbf{M}^{(d)}$; shaded regions are nonzero. Both diagonal blocks are ordinary causal masks, and all long-range structure lives in $\mathbf{G}$.

The construction of G. Let $d \geq 2$ and q be a prime. The geometry is the ambient space of $(d-1)$-dimensional vectors over the finite field $\mathbb{F}_q$, given by $\mathbb{F}_q^{d-1}$, with its affine hyperplanes $H_{a,b} = \{x : a^\top x = b\}$, one per normalized direction $a \neq 0$ and offset b, of which there are

$$B = \frac{q(q^{d-1} - 1)}{q - 1} = \Theta\big(q^{d-1}\big). \tag{5}$$

Each distant key is assigned a *profile*, a point of the ambient space, and each recent query a *type*, one of the hyperplanes,

$$\mathrm{prof} : [n] \to \mathbb{F}_q^{d-1}, \qquad \mathrm{type} : [m] \to \{0, \ldots, B-1\},$$

and a recent query attends to a distant key when that key's profile lies on the query's hyperplane:

$$\mathbf{G}_{ij} = \mathbb{1}\{\mathrm{prof}(j) \in H_{\mathrm{type}(i)}\}, \tag{6}$$

i.e. recent query of type h attends to its own prefix among the recent tokens and every distant token whose profile lies on H_h. $\mathbf{G}$ is the only component of the mask which is d-dependent and its structure dictates the VC-dimension (Theorem 3.1). In order to establish a VC lower bound for Theorem 3.1(iii), we further characterize prof and type. Let $e_1, \ldots, e_{d-1}$ be the standard basis of $\mathbb{F}_q^{d-1}$ and, for $R \subseteq [d-1]$, let H_R be the witness hyperplane. We impose the condition that there are distinct distant positions $j_1, \ldots, j_{d-1}$ with $\mathrm{prof}(j_\ell) = e_\ell$, and a recent index τ such that, for every $R \subseteq [d-1]$, H_R occurs at indices $i_R^- < \tau \le i_R^+$. We give two concrete examples:

1. The *positional* assignment takes $\mathrm{prof}(j)$ to be the base-q digits of $(j-1) \bmod q^{d-1}$ and $\mathrm{type}(i) = i \bmod B$.
2. The *content-based* assignment fixes a hash x which maps each token to a point of $\mathbb{F}_q^{d-1}$. This mapping can be fixed or learned. Let u_t be the hashed vector at position t. A distant key has $\mathrm{prof}(j) = u_j$ and for a hashed direction a, a recent query has $\mathrm{type}(i) = H_{a, a^\top u_{n+i}}$ (the hyperplane through its own cell). A query sees every distant key whose token repeats its own.

Scaling with sequence length. We define the family by prescribing the field size as a function of sequence length and VC-dimension. For each fixed $d \ge 2$, we choose a prime $q = \Theta(n^{1/d})$, where $n, m = \Theta(T)$. Writing $P = q^{d-1}$ for the number of profile cells, this gives $P = \Theta(T^{1-1/d})$ and $B = \Theta(T^{1-1/d})$. For each fixed d, a sufficiently large T ensures $P \le n$ and $2B \le m$, as required by our positional construction. Moreover, for $d = 1$, when $\mathbb{F}_q^0$ is a single point, we use $B = 1$ and $H_0 = \mathbb{F}_q^0$. Then $\mathbf{G}$ is all ones and the block form becomes $\mathbf{M}^{(1)} = \mathbf{L}_T$, ordinary causal masking.

Theorem 3.1 (Properties of hard-routing SMat masks $\mathbf{M}^{(d)}$)**.** *Let $T \ge 2$ and $1 \le d < \lfloor \log_2 T \rfloor$. Then* (i) $\mathbf{M}^{(d)} \in \{0,1\}^{T \times T}$ *is causal with* $\mathbf{M}^{(d)}_{tt} = 1$ *for all t;* (ii) $\mathbf{M}^{(1)} = \mathbf{L}_T$, (iii) $\mathrm{VC}(\mathbf{M}^{(d)}) = d$*; and* (iv) *the number of nonzero entries of $\mathbf{M}^{(d)}$, given by* $\mathrm{nnz}(\mathbf{M}^{(d)})$*, satisfies* $\mathrm{nnz}(\mathbf{M}^{(d)}) = \Theta(T^2)$.

3.1 Chunkwise SMat-attention

Splitting the mask. Split the block form into

$$\mathbf{M}^{(d)} = \underbrace{\begin{pmatrix} \mathbf{L}_n & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_m \end{pmatrix}}_{\text{two independent causal masks}} + \underbrace{\begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{G} & \mathbf{0} \end{pmatrix}}_{\text{long range}}. \tag{7}$$

Since $\bar{y}_i$ is linear in the mask, the two attention branches can be computed separately and added before the final division. By construction $\mathbf{G}_{ij}$ depends on i only through $\mathrm{type}(i)$ and on j only through $\mathrm{prof}(j)$. It therefore factors through the point–hyperplane incidence matrix

$$\mathbf{C} \in \{0,1\}^{B \times q^{d-1}}, \qquad \mathbf{C}_{hx} = \mathbb{1}\{x \in H_h\}, \qquad \mathbf{G}_{ij} = \mathbf{C}_{\mathrm{type}(i), \mathrm{prof}(j)}. \tag{8}$$

Writing $\mathbf{E} \in \{0,1\}^{n \times q^{d-1}}$ and $\mathbf{R} \in \{0,1\}^{m \times B}$ for the one-hot matrices of prof and type, $\mathbf{G} = \mathbf{R}\,\mathbf{C}\,\mathbf{E}^\top$. Distant keys sharing a profile can be pooled once and reused by every query that sees that profile, and recent queries sharing a type can be answered from one aggregated state.

The long-range branch. We apply the factorization $\mathbf{G} = \mathbf{R}\,\mathbf{C}\,\mathbf{E}^\top$ one factor at a time. $\mathbf{E}^\top$ pools the distant states into a *profile table* $\{F_x\}$, one state per profile; $\mathbf{C}$ converts this into a *type table* $\{U_h\}$, one state per query type; $\mathbf{R}$ contracts each query against the entry for its own type:

$$F_x := \sum_{j \le n:\, \mathrm{prof}(j) = x} \mathbf{Z}_j, \qquad U_h := \sum_x \mathbf{C}_{hx} F_x, \tag{9}$$

where $\bar{y}^{\mathrm{lr}}_{n+i} = \phi_{n+i}^\top U_{\mathrm{type}(i)}$ for $i \in [m]$, and $\bar{y}^{\mathrm{lr}}_i = 0$ for $i \le n$. Queries of a common type are contracted together in one matrix product and their rows scattered back into chronological order.

The causal branch. Following standard chunkwise formulations of linear attention (Hua et al., 2022; Beck et al., 2025; Yang et al., 2024a), partition $[n]$ and $(n, T]$ separately into N_c chronological chunks $I_1, \ldots, I_{N_c}$ of width at most c; Set $n = c\lfloor T/2c \rfloor$, so no chunk crosses block boundaries. Each chunk forms its summary $D_b = \sum_{j \in I_b} \mathbf{Z}_j$, an exclusive prefix scan (Blelloch, 1990; Yau et al.,

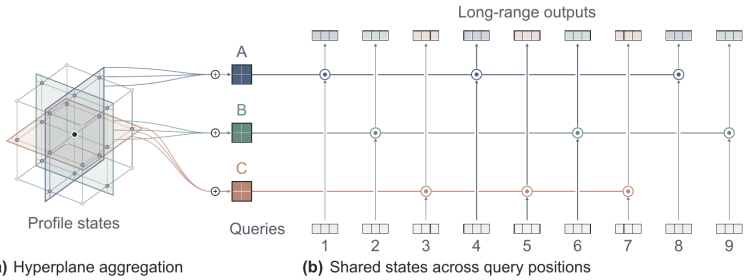

Figure 2: Hyperplane aggregation and shared-state reads in SMat-Attention. **(a)** A profile table is indexed by $\mathbb{F}_3^3$. Three coordinate hyperplanes each aggregate their incident profile states into a type state (A-C). The central profile belongs to all hyperplanes, so its state contributes to all the sums. **(b)** Each type state supplies every query of that type. The horizontal lines carry fixed shared states throughout these reads. The output scatters its results back to this order. The figure shows augmented long-range contributions, before causal addition and final normalization.

2025) over chunks gives the incoming state $S_b^{\text{in}} = \sum_{a<b} D_a$, and all chunks then run in parallel:

$$\bar{y}_i^{\text{loc}} = \underbrace{\phi_i^\top S_b^{\text{in}}}_{\text{earlier chunks}} + \underbrace{\phi_i^\top \sum_{j \in I_b,\, j \le i} \mathbf{Z}_j}_{\text{within chunk } b}, \qquad \bar{y}_i = \bar{y}_i^{\text{lr}} + \bar{y}_i^{\text{loc}}, \qquad i \in I_b, \tag{10}$$

followed by a row-wise division.

Algorithm 1 SMat forward pass for one attention head

Require: $\mathbf{Q}, \mathbf{K}, \mathbf{V}$; maps $\mathrm{prof}, \mathrm{type}$; incidence $\mathbf{C}$; chunk width c; $n = c\lfloor T/2c \rfloor$

1: Compute ϕ_i and $\mathbf{Z}_j$ as in Equation 2.
2: Pool $F_x = \sum_{\mathrm{prof}(j)=x} \mathbf{Z}_j$ over $j \le n$ by grouped reduction.
3: Tabulate $U_h = \sum_x \mathbf{C}_{hx} F_x$ for every type $h \in \{0, \dots, B-1\}$.
4: **for all** types h **in parallel**: contract all queries of type h against U_h and scatter the rows into chronological order.
5: **for all** chunks b **in parallel**: $D_b = \sum_{j \in I_b} \mathbf{Z}_j$.
6: Exclusive prefix scan over chunks, separately within $[n]$ and within $(n, T]$.
7: **for all** chunks b **in parallel**: evaluate Equation 10, add the long-range term, and normalize once.

Implicit incidence multiplication. We compute the type table $U = \mathbf{C}F$ without materializing $\mathbf{C}$. Let $\mathcal{A}$ contain the normalized nonzero directions in $\mathbb{F}_q^{d-1}$, with the first nonzero coordinate of each a equal to one. Index each type by its defining pair (a, b), using the same ordering as type. For a fixed direction a, every point x lies in exactly one hyperplane $H_{a,b}$, namely the one with $b = a^\top x$. Algorithm 2 therefore groups the profile states by this offset and accumulates $U_{a,b} = \sum_{x:\, a^\top x = b} F_x$, where the offsets use finite-field arithmetic and the state additions use ordinary arithmetic in $\mathbb{R}^{r \times p}$.

Algorithm 2 Implicit incidence tabulation

Require: Profile states $\{F_x\}$, field size q, dimension d

if $d = 1$ **then** $U_0 \leftarrow F_0$, and **return** U
for $a \in \mathcal{A}$ **in parallel do**
  $U_{a,b} \leftarrow 0_{r \times p}$ for all $b \in \mathbb{F}_q$
  for $x \in \mathbb{F}_q^{d-1}$ **do**
    $b \leftarrow a^\top x$ `/* Finite-field arithmetic`
    $U_{a,b} \leftarrow U_{a,b} + F_x$ `/* Ordinary state addition`
return U

For $d \geq 2$, the algorithm performs

$$|\mathcal{A}|q^{d-1} = \frac{q^{d-1}-1}{q-1}\, q^{d-1} = Bq^{d-2} = \Theta(q^{2d-3}) \tag{11}$$

state additions: one for each nonzero of $\mathbf{C}$. We enumerate points with a base-q counter and maintain $a^\top x$ as its entries change. Over a full traversal, the counter changes $O(q^{d-1})$ entries, so generating the offsets takes $O(|\mathcal{A}|q^{d-1})$ field operations. So, the total work is $O(q^{2d-3}rp)$, and the profile and type tables occupy $O((q^{d-1}+B)rp) = O(q^{d-1}rp)$ words. For $q = \Theta(T^{1/d})$, these bounds become $O(T^{2-3/d}rp)$ work and $O(T^{1-1/d}rp)$ words. If $d = 1$, we directly use the single profile state.

Theorem 3.2 (Chunkwise SMat-attention)**.** *Fix $d \geq 1$, and let $\mathbf{M}^{(d)}$ be the mask of Section 3 under the prescribed sequence-length scaling. Assume the kernel has a nonnegative feature factorization of dimension r, and every normalizer $\bar{y}_{i,p}$ is positive. Then, for large T, Algorithm 1 computes the attention outputs $o_1, \ldots, o_T$, in*

$$O\big((T^{2-3/d}+T)\,rp \;+\; T\,c\,(r+p)\big) \tag{12}$$

work, excluding evaluation of the feature maps, using $O\big((T^{1-1/d}+T/c)rp+c^2\big)$ words of working memory. The backward pass has the same asymptotic cost.

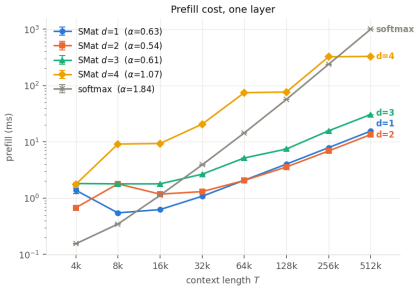

Figure 3: Prefill cost against fused causal attention (bf16, 8 heads, $r = 64$, chunk 128; median of 15 timed runs).

The proof of Theorem 3.2 is in Appendix E.

3.2 Gating

Recent variants of linear attention, such as gated linear attention (Yang et al., 2024a) and Mamba-2 (Dao and Gu, 2024), yield increased performance by weighting each edge according to the product of the gates between its source and target. To integrate our binary mask $\mathbf{M}^{(d)}$ into these continuous layers, we introduce a gating mechanism. Let $a_1, \ldots, a_T \in (0, 1]$ be per-token scalar decay gates, and $\lambda_i = \sum_{k \leq i} \log a_k$. We replace the binary mask $\mathbf{M}^{(d)}$ with the continuous mask $\widetilde{\mathbf{M}}^{(d)}$, given by:

$$\widetilde{\mathbf{M}}^{(d)}_{ij} = \mathbf{M}^{(d)}_{ij} \exp(\lambda_i - \lambda_j), \quad j \leq i \tag{13}$$

Note that the write scale of these layers is the key gate h_j from equation 2, which is already subsumed within $\mathbf{Z}_j$, and that the entries of $\widetilde{\mathbf{M}}^{(d)}$ are non-negative with the same support as $\mathbf{M}^{(d)}$. We show this additional gating does not affect training or decoding complexity in Lemma 4.

3.3 Memory-efficient decoding

For SMat-Attention with hard-routing, we compress the distant block once and reuse its summaries throughout the decoding process. For this, we begin by fixing the distance/recent boundary n in advance. After processing the distant tokens, we tabulate the type states $\{U_h\}_{h<B}$ and discard the intermediate profile states $\{F_x\}$. At each subsequent step $t > n$, we update a single running state $S_t = S_{t-1} + Z_t$, initialized with $S_n = 0$, and contract the query feature ϕ_t against $U_{\text{type}}(t-n) + S_t$ before normalization. Figure 4 illustrates this separation between fixed distant memory and evolving recent memory. The resulting decoder retains $B + 1$ states and performs $O(rp)$ state-update and readout work per token after tabulation. Appendix F relaxes the fixed boundary, extending the construction to decoding at any length without knowing T in advance.

Theorem 3.3 (Streaming decoding)**.** *Assume the hypotheses of Theorem 3.2, and suppose the total sequence length T is fixed in advance so that the distant/recent boundary n is known. Once Algorithm 1 has been run over the distant block, each subsequent token can be decoded in $O(rp)$ time, independent of T, from a cache of $O(T^{1-1/d}rp)$ words.*

We provide the proof of Theorem 3.3 in Appendix F.

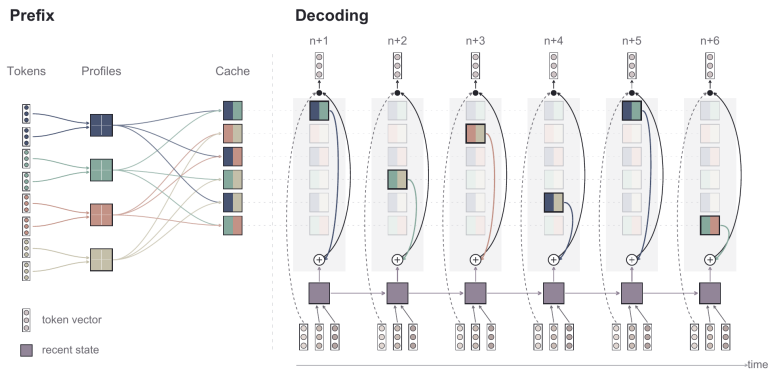

Figure 4: Cached decoding for additive SMat-Attention with hard-routing. **Left:** distant key-value contributions pool into profile states, which are added via the incidence structure to form the type cache. Colors identify the contributing profiles within each cached summary. **Right:** successive decoding steps access the fixed cache while maintaining one running state for recent tokens. The input vectors at each step are the query, key, and value. The key and value update the running state before answering the query. Repeated cache banks depict successive views of the shared memory.

3.4 Recurrent architectures and learned routing

We extend SMat's profile organization to recurrent sequence models. The local branch processes the distant and recent blocks separately, resetting its recurrent state and short convolution at the boundary. Distant tokens update profile memories F_x (specified in Appendix G), which are aggregated into type summaries $U_h = \sum_x C_{hx} F_x$. For a recent query, the memory contribution is combined with the local output before backbone normalization and output projection $z_i = z_i^{\mathrm{local}} + \lambda_i \widetilde{q}_i^{\top} \sum_h R_{ih} U_h$, where R_{ih} assigns queries to types and λ_i controls the memory contribution. To combine information across subsets, we introduce an independent query-dependent scorer over types. Each query selects the four highest-scoring types per head and applies a softmax over their scores to obtain the read weights R_{ih}, retaining a fixed number of summary reads.

The learned selector remains subquadratic because each query scores a sublinear number of cached types, $B = \Theta(T^{1-1/d})$, rather than all tokens. Across the sequence, scoring costs $\Theta(TB) = \Theta(T^{2-1/d})$, a factor of $T^{1/d}$ fewer scores than token-level all-pairs scoring. Type-summary construction is also subquadratic: Algorithm 2 costs $O(T^{2-3/d} rp)$, while a dense contraction costs $O(T^{2-2/d} rp)$. These routing and aggregation costs are subquadratic for every fixed finite $d \geq 2$ and fixed model dimensions. The linear-time bound for $d \leq 3$ applies to the hard-routing algorithm; learned selection instead incurs the subquadratic scoring cost above.

For learned content routing, a token's write address is computed from its current and preceding hidden states, while each recent query selects summaries using only its own causal representation. A learned projection quantizes the write representation into a discrete profile in $\mathbb{F}_q^{d-1}$. We train this hash jointly with the backbone using a straight-through gradient estimator and an auxiliary load-balancing loss. The four-read selector is learned separately through the softmax weights of its selected types. Thus, the model learns both where to store information and which summaries to retrieve; Appendix G.1 gives the details. We show in Appendix D that the VC-dimension of the support of the learned mask continues to be $O(d)$.

4 Experiments

We evaluate SMat on controlled synthetic tasks designed to probe routing and retrieval, followed by long-context language modeling.

Implementation and training details. We implement our models in PyTorch with custom Triton kernels (Tillet et al., 2019), and our experiments are run on single NVIDIA A100 GPUs. We use the Zoology training pipeline for MQAR (Arora et al., 2023), and AdamW with cosine learning-rate

decay. MQAR, joint context–key recall, and the 750M-token PG-19 experiments use the learned four-read routing variant of Section 3.4. Subset routing and multi-key retrieval use the mask-based constructions. All SMat masks were gated in the recurrent section of the mask. We provide our code* and defer details to Appendix G.

Subset Routing. We first test whether models using the proposed mask can realize the subset-selection patterns predicted by its VC dimension. Let $J = \{j_1, \ldots, j_k\}$ denote k landmark positions in the context. Each landmark j_ℓ stores an independent random payload x_ℓ in a distinct output channel, $v_{j_\ell} = x_\ell e_\ell$, while all non-landmark values are zero. A query specifies a subset $A \subseteq J$, and the target is $y_A = \sum_{j_\ell \in A} x_\ell e_\ell$. Payloads and requested subsets are resampled across examples, preventing the model from memorizing fixed input-output mappings.

This task tests the access patterns characterized by VC dimension. If k landmarks are shattered by the row supports of $\mathbf{M}$ with VC-dimension d, then every subset of those landmarks can be selected by some query and $\mathbf{M}$ can realize all subset-selection patterns over some d landmarks, but not over any $d+1$ landmarks. So, we expect performance to degrade once the requested routing dimension exceeds d. The task captures a basic requirement of long-context retrieval: selecting several relevant pieces of information while ignoring other nearby or similar context.

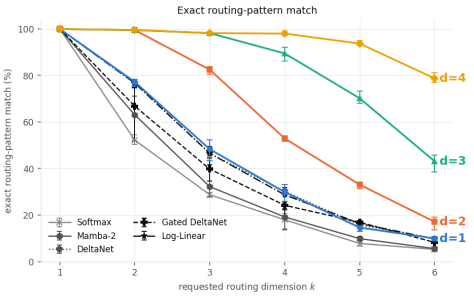

Figure 5: Routing-pattern match against the requested dimension k, after training with the mask fixed, T = 1024, mean of three seeds.

Multi-key retrieval. We next test content-based retrieval when a single query must recover multiple items. We randomly place N key-payload pairs throughout the distant context. Each query specifies k target keys and must retrieve the corresponding payloads. Since the locations of the pairs vary across examples, the model cannot solve the task using fixed positional routing. For SMat, any set of at most $d-1$ target profiles in $\mathbb{F}_q^{d-1}$ lies on a common affine hyperplane. This guarantees that the target profiles lie on a common hyperplane, but does not guarantee exact selection: distractors or profile collisions may also lie on that hyperplane. We report exact-support accuracy (Appendix G.2), where a query is 'correct' only if every requested item is present in the output and every unrequested item is absent.

Model	$k=1$	$k=2$	$k=3$
Softmax	44.29 (49.00)	66.60 (57.68)	66.46 (57.55)
Linear Baselines	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
LSH bucketing (125 buckets)	100.00 (0.00)	1.71 (0.50)	0.03 (0.03)
SMat ($d=2$)	100.00 (0.00)	13.90 (2.13)	1.29 (0.33)
SMat ($d=3$)	100.00 (0.00)	99.82 (0.10)	34.20 (1.21)
SMat ($d=4$)	99.98 (0.03)	99.92 (0.03)	98.99 (0.03)

Table 1: Exact routing-pattern match (%) on multi-key retrieval, by the number k of marked positions. Payloads are resampled each batch and evaluation uses fresh ones. Mean (std) over 3 seeds. SMat performs better at higher k as d increases.

Multi-query associative recall. We train on the Zoology MQAR (Arora et al., 2023) mixture with 4-64 key-value pairs at sequence lengths 64–256, using two-layer models with head and state dimension 16. Evaluation uses 1000 held-out examples per load at sequence lengths 64-256. Accuracy is averaged over query tokens within each example, then equally over examples, giving equal weight to each load. Log-Linear Attention (Guo et al., 2025) uses the same training harness and budget as the corresponding backbone. Table 2 reports retrieval accuracy for Mamba-2 and Gated DeltaNet with and without SMat. At every width, for both backbones, the SMat variants perform well.

Joint Context–Key Recall We adapt the multi-query joint recall task of Zhan et al. (2025), which requires retrieving values using both a context and a key. We represent each binding as an explicit $(\text{context}, \text{key}, \text{value})$ record and independently shuffle the records and queries. Keys repeat across contexts, while values are sampled independently, requiring joint identification of the requested

*https://anonymous.4open.science/r/smat_attention/README.md

Table 2: Final MQAR accuracy (%), 32 epochs. Mean (std) over 3 seeds.

Backbone	Width	Base	Log-Linear	$d = 2$	$d = 3$	$d = 4$
Gated DeltaNet	16	44.34 (7.58)	44.17 (10.90)	49.82 (5.21)	45.40 (3.70)	44.63 (1.39)
Gated DeltaNet	32	63.99 (4.14)	68.41 (8.13)	70.07 (0.48)	79.74 (2.65)	75.21 (6.78)
Gated DeltaNet	64	70.59 (7.96)	79.06 (2.11)	78.88 (8.01)	87.37 (4.12)	80.52 (6.07)
Mamba-2	16	41.36 (3.07)	46.47 (7.61)	47.61 (4.36)	61.57 (2.41)	49.19 (5.93)
Mamba-2	32	73.55 (5.02)	76.57 (4.90)	75.44 (0.19)	77.89 (4.45)	81.90 (2.49)
Mamba-2	64	87.87 (4.26)	85.85 (1.92)	91.76 (0.95)	92.89 (2.13)	93.14 (1.55)

record. We train on 180K examples spanning 4, 16, 128, 256, and 512 bindings, using two-layer, width-64 models for 32 epochs. Table 3 reports validation accuracy averaged across the five memory loads. Among the tested SMat settings, $d = 3$ achieves the highest mean accuracy and lowest sample standard deviation on both backbones. It outperforms the native backbones in mean accuracy, while scoring above Log-Linear on GDN and below it on Mamba-2. These results do not establish the cause of the differences across d.

Variant	Mamba-2	GDN
Native	51.14 (17.74)	53.78 (2.85)
+ Log-Linear	66.01 (19.88)	57.78 (5.36)
+ SMat ($d = 2$)	54.10 (3.72)	51.25 (5.28)
+ SMat ($d = 3$)	61.65 (3.15)	60.15 (3.27)
+ SMat ($d = 4$)	59.12 (15.12)	59.89 (9.34)

Table 3: Average validation accuracy (%) on joint context–key recall, averaged over five memory loads. All models use width 64, 32 epochs, learning rate 0.003. Mean (std) over 3 seeds.

Natural Language Modeling. We use language modeling to evaluate whether the additional routing structure from SMat preserves the modeling capabilities of the underlying architectures. We evaluate language modeling on PG-19 (Rae et al., 2019) using the GPT-2 tokenizer. We train separate models at context lengths of 16K and 32K, each on 300M tokens, and report per-token loss on held-out tests at the corresponding training context length. All models use 8 layers, a hidden width of 384, and the same training data order. SMat, Mamba-2, and the Transformer have 27.8M, 26.9M, and 33.5M parameters, respectively. Across both context lengths, SMat variants achieve slightly lower negative log-likelihood (NLL) than the similarly-sized Mamba-2 baseline, with small differences among $d = 2, 3, 4$, as seen in Table 16a in Appendix G.

We further evaluate SMat augmentation of GDN and Mamba-2 at a larger scale, using 16-layer models with hidden width 768, a 16K context length, and 750M training tokens. As shown in 16b, all SMat variants achieve roughly the same perplexity as its baselines, suggesting that SMat's performance in language-modeling is comparable across both backbone architectures in this setting.

5 Conclusion and Future Work

We introduced SMat-Attention, a framework for explicitly trading off the flexibility of long-range access against computation and memory. Rather than compressing the entire past into a fixed-size state or allowing unrestricted token-level interactions, SMat-Attention provides an intermediate regime in which the richness of long-range routing is controlled by a single parameter, the VC dimension d. This structure yields provably subquadratic training and constant-time decoding for the hard-routing masks, and the learned extensions inherit those bounds up to a subquadratic selector term. Controlled routing experiments show benefits consistent with this expressiveness. Learned SMat extensions improve mean associative and joint context–key recall accuracy in several tested settings, while remaining comparable in performance at small-scale language modeling.

We focus on a structured finite-feature setting and a fixed-horizon decoding formulation, while some empirical variants introduce additional learned routing and memory updates. Extending the framework to more adaptive routing schemes, dynamic contexts, and larger-scale language models is a natural direction for future work. Our results suggest that explicitly controlling the complexity of long-range access patterns is a useful way to navigate tradeoffs between expressivity and computation.

AI USE STATEMENT

In this work, we used generative AI tools for literature searches, coding implementation, and to aid in the presentation of our experimental results. We also used AI assistance to edit the manuscript, help design scientific figures, and validate our mathematical claims. The authors take full responsibility for verifying the correctness and originality of all the material in the manuscript, including the theoretical claims, experimental results, code, and figures.

ETHICS STATEMENT

In this work, we investigate the computational properties of the attention mechanism used in the transformer, and study tradeoffs between the memory, expressiveness, and computation across various representations. Our evaluation uses synthetic tasks and Google DeepMind's PG-19 corpus for measuring the ability of language models to process long-range contexts. In general, efficiency gains in this line of work may broaden the access to long-context modeling, while also lowering the cost of processing potentially sensitive pieces of textual information. Moreover, models using the proposed mechanism are subject to the standard bias, privacy, and misuse risks associated with language models, and the attention mechanism on its own does not provide safeguards against these risks. We discuss broader impacts in Appendix A.

REPRODUCIBILITY STATEMENT

Section 3 specifies the mask construction and attention algorithms, including pseudocode for the forward pass and implicit incidence tabulation. We provide proofs of our theoretical claims in Appendices C-F. Section G describes the experimental tasks, datasets and synthetic-data generation procedures, model configurations, optimization settings, training budgets, evaluation protocols, and computing hardware. Finally, we provide an anonymous link to a faithful implementation of our code in the main body of our paper, along with documentation on how to run the experiments.

REFERENCES

Emile Anand, Jan van den Brand, and Rose McCarty. The structural complexity of matrix-vector multiplication, 2025. URL https://arxiv.org/abs/2502.21240.

Emile Anand, Abdullah Ateyeh, Xinyuan Cao, and Max Dabagia. Continuous latent contexts enable efficient online learning in transformers, 2026. URL https://arxiv.org/abs/2605.09867.

Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Zou, Atri Rudra, and Christopher Ré. Zoology: Measuring and improving recall in efficient language models, 2023. URL https://arxiv.org/abs/2312.04927.

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, Dylan Zinsley, James Zou, Atri Rudra, and Christopher Ré. Simple linear attention language models balance the recall-throughput tradeoff. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*. PMLR, 2024.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014. URL https://arxiv.org/abs/1409.0473.

Maximilian Beck, Korbinian Pöppel, Phillip Lippe, and Sepp Hochreiter. Tiled flash linear attention: More efficient linear rnn and xlstm kernels. *Advances in Neural Information Processing Systems*, 38:75093–75148, 2025.

Ali Behrouz, Zeman Li, Yuan Deng, Peilin Zhong, Meisam Razaviyayn, and Vahab Mirrokni. Memory caching: RNNs with growing memory. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.24281.

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

Guy E. Blelloch. Prefix sums and their applications. Technical Report CMU-CS-90-190, Carnegie Mellon University, Department of Computer Science, November 1990. URL https://www.cs.cmu.edu/~scandal/papers/CMU-CS-90-190.html.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers, 2022. URL https://arxiv.org/abs/2009.14794.

Krzysztof Choromanski, Han Lin, Haoxian Chen, Tianyi Zhang, Arijit Sehanobish, Valerii Likhosherstov, Jack Parker-Holder, Tamas Sarlos, Adrian Weller, and Thomas Weingarten. From block-toeplitz matrices to differential equations on graphs: towards a general theory for scalable masked transformers, 2023. URL https://arxiv.org/abs/2107.07999.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning, 2023. URL https://arxiv.org/abs/2307.08691.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized Models and Efficient Algorithms Through Structured State Space Duality. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 10041–10071. PMLR, 2024.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness, 2022. URL https://arxiv.org/abs/2205.14135.

Jusen Du, Weigao Sun, Disen Lan, Jiaxi Hu, Tao Zhang, and Yu Cheng. MoM: Linear sequence modeling with mixture-of-memories. In *The Fourteenth International Conference on Learning Representations*, 2026.

Jacob Fein-Ashley, Neelesh Gupta, Rajgopal Kannan, and Viktor Prasanna. Spectre: An fft-based efficient drop-in replacement to self-attention for long contexts, 2025. URL https://arxiv.org/abs/2502.18394.

Daniel Y. Fu, Tri Dao, Khaled K. Saab, Armin W. Thomas, Atri Rudra, and Christopher Ré. Hungry hungry hippos: Towards language modeling with state space models, 2023. URL https://arxiv.org/abs/2212.14052.

Daniel Goldstein, Navneel Singhal, and Eugene Cheah. Key-value means: Transformers with expandable block-recurrent compressed memory, 2026. URL https://arxiv.org/abs/2605.09877. arXiv preprint.

Albert Gu and Tri Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces, 2022. URL https://arxiv.org/abs/2111.00396.

Han Guo, Songlin Yang, Tarushii Goel, Eric P. Xing, Tri Dao, and Yoon Kim. Log-linear attention, 2025. URL https://arxiv.org/abs/2506.04761.

Weizhe Hua, Zihang Dai, Hanxiao Liu, and Quoc V. Le. Transformer quality in linear time, 2022. URL https://arxiv.org/abs/2202.10447.

Qing-Yuan Jiang, Xue Cui, and Wu-Jun Li. Deep discrete supervised hashing. *IEEE Transactions on Image Processing*, 27(12):5996–6009, December 2018. ISSN 1941-0042. doi: 10.1109/tip.2018.2864894.

Praneeth Kacham, Vahab Mirrokni, and Peilin Zhong. Polysketchformer: Fast transformers via sketching polynomial kernels. *arXiv preprint arXiv:2310.01655*, 2023.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention, 2020. URL https://arxiv.org/abs/2006.16236.

Tobias Katsch. Gateloop: Fully data-controlled linear recurrence for sequence modeling, 2024. URL https://arxiv.org/abs/2311.01927.

Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, Cambridge, MA, USA, 1994. ISBN 978-0-262-11193-5.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer, 2020. URL https://arxiv.org/abs/2001.04451.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626, 2023.

Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems*, 32, 2019.

Hao Liu, Matei Zaharia, and Pieter Abbeel. Ring attention with blockwise transformers for near-infinite context. In *International Conference on Learning Representations*, 2024.

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, Zhiqi Huang, Huan Yuan, Suting Xu, Xinran Xu, Guokun Lai, Yanru Chen, Huabin Zheng, Junjie Yan, Jianlin Su, Yuxin Wu, Neo Y. Zhang, Zhilin Yang, Xinyu Zhou, Mingxing Zhang, and Jiezhong Qiu. MoBA: Mixture of block attention for long-context LLMs, 2025. URL https://arxiv.org/abs/2502.13189.

Shengjie Luo, Shanda Li, Tianle Cai, Di He, Dinglan Peng, Shuxin Zheng, Guolin Ke, Liwei Wang, and Tie-Yan Liu. Stable, fast and accurate: Kernelized attention with relative positional encoding. *Advances in Neural Information Processing Systems*, 34:22795–22807, 2021.

Stefano Massaroli, Michael Poli, Daniel Y. Fu, Hermann Kumbong, Rom N. Parnichkun, Aman Timalsina, David W. Romero, Quinn McIntyre, Beidi Chen, Atri Rudra, Ce Zhang, Christopher Re, Stefano Ermon, and Yoshua Bengio. Laughing hyena distillery: Extracting compact recurrences from convolutions, 2023. URL https://arxiv.org/abs/2310.18780.

Bo Peng, Daniel Goldstein, Quentin Anthony, Alon Albalak, Eric Alcaide, Stella Biderman, Eugene Cheah, Xingjian Du, Teddy Ferdinan, Haowen Hou, Przemysław Kazienko, Kranthi Kiran GV, Jan Kocoń, Bartłomiej Koptyra, Satyapriya Krishna, Ronald McClelland Jr., Jiaju Lin, Niklas Muennighoff, Fares Obeid, Atsushi Saito, Guangyu Song, Haoqin Tu, Cahya Wirawan, Stanisław Woźniak, Ruichong Zhang, Bingchen Zhao, Qihang Zhao, Peng Zhou, Jian Zhu, and Rui-Jie Zhu. Eagle and finch: Rwkv with matrix-valued states and dynamic recurrence, 2024. URL https://arxiv.org/abs/2404.05892.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A. Smith, and Lingpeng Kong. Random feature attention, 2021. URL https://arxiv.org/abs/2103.02143.

Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y. Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Re. Hyena hierarchy: Towards larger convolutional language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 28043–28078. PMLR, 2023.

Zhen Qin, Xiaodong Han, Weixuan Sun, Bowen He, Dong Li, Dongxu Li, Yuchao Dai, Lingpeng Kong, and Yiran Zhong. Toeplitz neural network for sequence modeling, 2023. URL https://arxiv.org/abs/2305.04749.

Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. Hgrn2: Gated linear rnns with state expansion, 2024. URL https://arxiv.org/abs/2404.07904.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling, 2019. URL https://arxiv.org/abs/1911.05507.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers, 2020. URL https://arxiv.org/abs/2003.05997.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9355–9366. PMLR, 2021.

Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 01 1992. ISSN 0899-7667. doi: 10.1162/neco.1992.4.1.131. URL https://doi.org/10.1162/neco.1992.4.1.131.

Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. FlashAttention-3: Fast and accurate attention with asynchrony and low-precision. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-2193.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models, 2023. URL https://arxiv.org/abs/2307.08621.

Philippe Tillet, Hsiang-Tsung Kung, and David Cox. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pages 10–19, 2019.

Vladimir N. Vapnik and Alexey Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and Its Applications*, 16(2):264–280, 1971. doi: 10.1137/1116025.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. URL https://arxiv.org/abs/1706.03762.

Bernard Widrow and Marcian E. Hoff. Adaptive switching circuits. In *1960 IRE WESCON Convention Record*, volume 4, pages 96–104. Institute of Radio Engineers, 1960.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 56501–56523. PMLR, 2024a.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In *Advances in Neural Information Processing Systems*, volume 37, 2024b.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule, 2025. URL https://arxiv.org/abs/2412.06464.

Morris Yau, Sharut Gupta, Valerie Engelmayer, Kazuki Irie, Stefanie Jegelka, and Jacob Andreas. Sequential-parallel duality in prefix scannable models. arXiv:2506.10918, 2025.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Y. X. Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention, 2025. URL https://arxiv.org/abs/2502.11089.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: transformers for longer sequences. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Zhihao Zhan, Jianan Zhao, Zhaocheng Zhu, and Jian Tang. Overcoming long context limitations of state space models via context dependent sparse attention. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/3e4ea825a66bf1e1e33eca3e81fe7886-Abstract-Conference.html.

Outline of the Appendices.

- Section A explains the broader societal impacts of our work,
- Section B lists the related work,
- Section C states an auxiliary lemma to motivate our kernel attention mechanism,
- Section D proves our theorem to characterize the properties of the masks,
- Section E proves the chunkwise SMat-attention result,
- Section F discusses extension to horizon-free decoding, and
- Section G lists the training details

A Broader Impacts

SMat-Attention targets the computational and memory costs of long-context sequence modeling. Namely, more efficient prefill and decoding could lower the energy and hardware requirements of deploying long-context models, potentially widening access to researchers and practitioners without large compute budgets. By making long-range access complexity an explicit architectural parameter, our framework also offers a more interpretable theory on which parts of the context a model can attend to, which may aid analysis of how long-context models retrieve and use information. At the same time, cheaper long-context inference lowers the barrier to processing large volumes of personal or sensitive text, and the general risks of language models, including the generation of misleading or harmful content, apply to systems built on this mechanism.

B Related Work

	prefill	per decoded token	decode cache
Softmax attention	$\Theta(T^2 d_v)$	$\Theta(T d_v)$	$\Theta(T)$ pairs
Linear attention	$\Theta(Trp)$	$\Theta(rp)$	1 state
Log-linear attention	$\Theta(T \log T\, rp)$	$\Theta(rp \log T)$	$\Theta(\log T)$ states
SMat, VC dim. d	$O((T^{2-3/d} + T)rp)$	$O(rp)$	$O(T^{1-1/d})$ states

Table 4: SMat sits between linear and softmax attention. It keeps the $O(rp)$ per-token decoding cost of linear attention and pays for VC dimension d in cache size rather than in decoding time.

Kernel and recurrent attention. Kernelized attention factorizes the content kernel as $K(q,k) = \phi_{\mathbf{Q}}(q)^\top \phi_{\mathbf{K}}(k)$, allowing causal attention to be accumulated in finite-dimensional recurrent states (Kacham et al., 2023). For fixed feature and value dimensions, this yields linear work in the sequence length, as well as a recurrent state whose size is independent of T (Katharopoulos et al., 2020). Random-feature methods such as Performer (Choromanski et al., 2022) approximate the softmax kernel within this framework. More recent architectures enrich the recurrent state and its update (Sun et al., 2023; Katsch, 2024; Qin et al., 2024; Peng et al., 2024): for instance, gated linear attention introduces input-dependent retention (Peng et al., 2021; Yang et al., 2024a), whereas DeltaNet uses key-conditioned corrective updates (Schlag et al., 2021; Yang et al., 2024b). These mechanisms improve how a fixed-size state is maintained, but retaining a sequence-length-independent state creates a capacity-recall tradeoff on tasks requiring access to many independent items from the context (Arora et al., 2024). Our work differs: SMat-Attention accepts any supplied finite nonnegative feature factorization and changes the *causal support pattern* over which the resulting features interact.

Structured sequence mixing and hierarchical memory. A recurring theme in efficient sequence modeling is that computational savings arise from algebraic structure in the sequence mixing matrix since many efficient sequence models can be interpreted as multiplication by a structured causal matrix. For instance, linear attention induces a lower-triangular structured operator (Katharopoulos et al., 2020), whereas long-convolution architectures such as Hyena (Massaroli et al., 2023; Poli et al., 2023) use Toeplitz-like operators (Qin et al., 2023; Fein-Ashley et al., 2025) with

FFT-based multiplication. Similarly, selective state-space models (SSMs) such as Mamba induce input-dependent semi-separable mixing matrices (Gu and Dao, 2023). Mamba-2 makes this matrix perspective explicit through structured state-space duality, showing an equivalence between SSM recurrences and multiplication by semi-separable matrices (Dao and Gu, 2024). More generally, Choromanski et al. (2023) showed that efficient multiplication by a mask can be lifted to efficient finite-feature masked attention, encompassing causal prefix masks, relative-position operators, and a variety of graph-derived masks. These examples suggest treating the structure of the sequence mixing matrix itself as a design space.

The most closely related work to ours is Log-Linear Attention, which replaces linear attention's single prefix state with $O(\log T)$ summaries of disjoint dyadic buckets maintained through a Fenwick-tree schedule (Guo et al., 2025). Its hierarchical matrix structure supports $O(T \log T)$ parallel training and $O(\log T)$ time and memory per decoded token, while query-dependent coefficients select among temporal scales. We explore a different structural axis in SMat-Attention by constructing overlapping binary access patterns from point-hyperplane incidences, and quantifying their combinatorial richness via the VC-dimension. Therefore, SMat Attention's states summarize geometric profiles rather than temporal buckets, and our resulting guarantee is different: we show that after a fixed, known distant prefix has been processed, SMat-Attention decodes each subsequent token in time independent of T using $O(T^{1-1/d})$ cached feature states.

VC dimension and structured matrix multiplication. Beyond sequence modeling, a line of work studies when structured matrices result in fast matrix-vector multiplication. The VC-dimension is a combinatorial complexity metric that classically measures the richness of a set system through the subsets it realizes (Kearns and Vazirani, 1994). Recent work connects this quantity to the complexity of matrix-vector multiplication. For instance, after an $\tilde{O}(T^2)$ preprocessing of a $T \times T$ Boolean matrix $\mathbf{M}$ of VC-dimension d, Anand et al. (2025) gives an algorithm for multiplying $\mathbf{M}$ by an arbitrary vector in $\widetilde{O}(T^{2-1/d})$ time. This connection motivates our use of VC dimension as an access-complexity measure; our construction additionally exploits point–hyperplane incidence structure to obtain a sharper specialized algorithm.

Sparse and hardware-efficient attention. Hardware-aware algorithms such as FlashAttention reorganize exact softmax attention into on-chip tiles, substantially reducing memory traffic without changing its worst-case quadratic arithmetic complexity (Dao et al., 2022). Sparse-attention methods instead reduce the number of evaluated query-key pairs through local and global windows, random edges, or content-dependent selection (Beltagy et al., 2020; Zaheer et al., 2020; Yuan et al., 2025; Li et al., 2019). Conversely, SMat-Attention's complete causal mask has $\Theta(T^2)$ non-zeros, and therefore cannot be evaluated efficiently by enumerating all permitted interactions. Although its long-range block contains $\Theta(T^{2-1/d})$ token-level edges, sparsity alone would only yield the corresponding $\Theta(T^{2-1/d})$ computation. Our sharper bound comes from additional reuse: distant keys with the same profile are pooled once, queries with the same type share an aggregate, and the two causal blocks are handled by prefix scans and local dense tiles. Thus, SMat-Attention combines hardware-friendly intrachunk computation with algebraic reuse across chunks; its speedup is not merely a consequence of deleting attention edges. Additionally, the Reformer's LSH attention (Kitaev et al., 2020) and the Routing Transformer (Roy et al., 2020) restrict each query to the keys sharing its hash or cluster, and our content-based assignment uses a similar device. The structures differ for $d \geq 3$: hash and cluster buckets partition the keys, so a query sees exactly one cell, whereas each of our hyperplanes contains q^{d-2} profiles, so a query reads a structured union of cells. For $d = 2$ the hyperplanes are single points and SMat reduces to bucketed linear attention.

Linear attention and its variants. Linear attention replaces the softmax kernel with a feature map that factorizes the attention matrix, allowing $O_t = \phi(q_t)^\top \sum_{j \leq t} \phi(k_j) v_j^\top$ to be computed using a fixed-size recurrent state. This reduces autoregressive decoding from linear to constant cost per token and allows for efficient parallel training, but compresses the entire history into a fixed-size state (Katharopoulos et al., 2020). However, vanilla linear attention compresses history into a fixed-size state by accumulating key–value associations, leading to memory-capacity limitations and interference between stored associations (Schlag et al., 2021). Gated variants augment this recurrence with input-dependent retention factors that modulate the existing state, allowing the model to selectively preserve or forget past information while retaining efficient recurrent inference. For example, gated linear attention updates the state as $S_t = A_t \odot S_{t-1} + v_t k_t^\top$, where the gate A_t determines which parts of the previous state are retained, while preserving their efficient recurrent

and parallel forms (Yang et al., 2024a). Gating improves memory management by controlling how much of the existing state is retained, but it does not directly account for what value is already stored at a particular key. Delta-rule models make the update key-specific by using the current prediction error, $S_t = S_{t-1}(I - \beta_t k_t k_t^\top) + \beta_t v_t k_t^\top$, so that writing at k_t partially removes the value currently associated with that key before inserting v_t (Yang et al., 2024b).

C Auxiliary Lemmas

Lemma 1. *Assume that the mask $\mathbf{M} \in \mathbb{R}^{T\times T}$ supports matrix-vector multiplication in time $f_{\mathbf{M}}(T)$. Then, the general masked kernel attention algorithm with mask $\mathbf{M}$ can be implemented in time $O((f_{\mathbf{M}}(T)+T)rd_v)$.*

Proof. Note that the i'th token representation obtained from the general masked kernel attention has the form

$$\mathsf{Att}_K(\mathbf{Q},\mathbf{K},\mathbf{V},\mathbf{M})_i = \frac{\phi(\mathbf{q}_i^\top)^\top \sum_{j=1}^T \mathbf{M}_{i,j}\phi(\mathbf{k}_j^\top)^\top v_j}{\phi(\mathbf{q}_i^\top)^\top \sum_{j=1}^T \mathbf{M}_{i,j}\phi(\mathbf{k}_j^\top)}.$$

Then, following Luo et al. (2021), let

$$\mathbf{H}^1 = \left(\sum_{j=1}^T \mathbf{M}_{i,j}\phi(\mathbf{k}_j^\top)\mathbf{v}_j\right)_{i=1}^T \in \mathbb{R}^{r\times d_v}$$

and

$$\mathbf{H}^2 = \left(\sum_{j=1}^T \mathbf{M}_{i,j}\phi(\mathbf{k}_j^\top)^\top\right)_{i=1}^T \in \mathbb{R}^{1\times r}.$$

Note that if $\mathbf{D}^1$ and $\mathbf{D}^2$ are the vectorized forms of $\mathbf{H}^1$ and $\mathbf{H}^2$ (respectively), where each element of the sequence is vectorized and the resulting vectors are stacked into matrices, then $\mathbf{D}^1 = \mathbf{M}\mathbf{V}^1$ and $\mathbf{D}^2 = \mathbf{M}\mathbf{V}^2$, where the i'th rows of $\mathbf{V}^1$ and $\mathbf{V}^2$ are given as $\mathbf{V}_i^1 = \mathrm{vec}(\phi(\mathbf{k}_i)^\top \mathbf{v}_i)$ and $\mathbf{V}_i^2 = \phi(\mathbf{k}_i^\top)^\top$. Therefore, computing $\mathbf{H}^1$ and $\mathbf{H}^2$ takes time $f_{\mathbf{M}}(T)rd_v$, and so Att_i can be computed in time $O((f_{\mathbf{M}}(T)+T)rd_v)$, which completes the proof. □

D VC dimension of the causal incidence masks

Lemma 2 (Lower triangular matrices)**.** *The $T\times T$ binary lower triangular matrix $\mathbf{L}_T$ has VC-dimension 0 for $T=1$ and 1 for $T\geq 2$.*

Proof. For $T=1$, the only row support is $\{1\}$, so no singleton is shattered.

For $T\geq 2$, note that column 2 is excluded by row 1 and included by row 2, and so the singleton $\{2\}$ is shattered, which proves $\mathrm{VC}(\mathbf{L}_T)\geq 1$. Next, take any two distinct column/row indices x, y, with $x<y$. In $\mathbf{L}_T$, the upper-right entries are all 0's. So, if a row indicator functions is 1 at a later index, we cannot have independent labelings. Therefore, since no configuration of two points can achieve all $2^2=4$ binary label combinations, the VC-dimension is less than 2, completing the proof. □

Lemma 3 (Affine hyperplanes)**.** *Let $D\geq 1$ and let q be a prime power. Then, the set system of all proper affine hyperplanes in $\mathbb{F}_q^D$ has VC dimension exactly D.*

Proof. We first record an affine-dependence observation. Suppose profiles $x_j \in \mathbb{F}_q^D$, indexed by a set C, satisfy

$$\sum_{j\in C}\lambda_j x_j = 0, \qquad \sum_{j\in C}\lambda_j = 0, \qquad \lambda_j \neq 0 \quad (j\in C). \tag{*}$$

For every $a \in C$, any affine hyperplane containing all x_j with $j \in C \setminus \{a\}$ also contains x_a: indeed,

$$x_a = -\sum_{j \in C\setminus\{a\}} \frac{\lambda_j}{\lambda_a} x_j, \qquad -\sum_{j \in C\setminus\{a\}} \frac{\lambda_j}{\lambda_a} = 1.$$

This observation also applies to repeated profiles at distinct indices.

If $D+1$ points were shattered, the all-included trace would place them in a proper hyperplane of affine dimension $D-1$. They are therefore affinely dependent. Restricting a nonzero dependence to its nonzero coefficients gives a set C satisfying $(*)$. The preceding observation rules out the trace $C \setminus \{a\}$, contradicting shattering. This proves the upper bound.

For the lower bound, let $e_1, \ldots, e_D$ be the standard basis. For each $\mathbf{R} \subseteq [D]$, define

$$H_R = \begin{cases} \left\{ x \in \mathbb{F}_q^D : \sum_{\ell \notin R} x_\ell = 0 \right\}, & R \neq [D], \\ \left\{ x \in \mathbb{F}_q^D : \sum_{\ell=1}^{D} x_\ell = 1 \right\}, & R = [D]. \end{cases} \tag{14}$$

Each defining normal is nonzero, so every H_R is a proper affine hyperplane. Moreover, $e_\ell \in H_R$ if and only if $\ell \in R$. Thus the standard basis is shattered. The construction is valid over every finite field, including $\mathbb{F}_2$, which completes the proof. □

Theorem 3.1 (Properties of hard-routing SMat masks $\mathbf{M}^{(d)}$)**.** *Let $T \geq 2$ and $1 \leq d < \lfloor \log_2 T \rfloor$. Then* (i) *$\mathbf{M}^{(d)} \in \{0,1\}^{T\times T}$ is causal with $\mathbf{M}^{(d)}_{tt} = 1$ for all t;* (ii) *$\mathbf{M}^{(1)} = \mathbf{L}_T$,* (iii) *$\mathrm{VC}(\mathbf{M}^{(d)}) = d$; and* (iv) *the number of nonzero entries of $\mathbf{M}^{(d)}$, given by $\mathrm{nnz}(\mathbf{M}^{(d)})$, satisfies $\mathrm{nnz}(\mathbf{M}^{(d)}) = \Theta(T^2)$.*

Proof. Let $\mathbf{M}^{(d)}$ be the mask constructed in Section 3. Let q be a prime. Then, for every $1 \leq d < \lfloor \log_2 T \rfloor$ for sufficiently large T, we prove the above properties.

Recall the block form from Section 3:

$$\mathbf{M}^{(d)} = \begin{pmatrix} \mathbf{L}_n & 0 \\ \mathbf{G} & \mathbf{L}_m \end{pmatrix}, \qquad \mathbf{G}_{ij} = \mathbf{1}\{\mathrm{prof}(j) \in H_{\mathrm{type}(i)}\}.$$

The two diagonal blocks are inclusive causal triangles, and every entry of $\mathbf{G}$ connects a recent query to a distant key. This proves causality, the unit diagonal, and $\mathbf{M}^{(d)} \leq \mathbf{L}_T$.

We prove (iii) and (iv) for every d. First, for $d = 1$, the mask is $\mathbf{L}_T$ which clearly satisfies (i) and (ii) and (iv). Then, from lemma 2, (iii) is satisfied. Moreover, note that

$$\mathrm{nnz}(\mathbf{M}^{(d)}) \geq \frac{n(n+1) + m(m+1)}{2} \geq \frac{(n+m)^2}{4} = \frac{T^2}{4},$$

while causality gives $\mathrm{nnz}(\mathbf{M}^{(d)}) \leq T(T+1)/2$.

Finally, it remains to prove the VC-dimension claim for $d \geq 2$. For this, let $D = d - 1$.

Upper bound. Suppose a set J of $d+1$ column indices is shattered. Split it into distant and recent indices, $J = J_{\mathrm{dist}} \sqcup J_{\mathrm{rec}}$. The traces on the recent block are empty for distant rows and prefixes for recent rows. In particular, if $u < v$ are recent indices, every row containing v also contains u. Thus $|J_{\mathrm{rec}}| \leq 1$. We proceed by casework:

If $|J_{\mathrm{rec}}| = 1$, consider the labelings in which that recent coordinate is one. Their realizing rows must be recent rows, whose distant supports are $\{j \in [n] : \mathrm{prof}(j) \in H_{\mathrm{type}(i)}\}$. Therefore, the $d = D + 1$ distant indices would be shattered by proper affine hyperplanes in $\mathbb{F}_q^D$. If two of these indices share a profile, they cannot be independently labeled. Otherwise, their profiles would form a shattered set of $D+1$ points, contradicting Lemma 3. Hence this case is impossible.

On the other hand, if $|J_{\mathrm{rec}}| = 0$, all $d + 1 = D + 2$ indices are distant. Their profiles are affinely dependent, so there exist a set $C \subseteq J$ and coefficients satisfying

$$\sum_{j \in C} \lambda_j \,\mathrm{prof}(j) = 0, \qquad \sum_{j \in C} \lambda_j = 0, \qquad \lambda_j \neq 0 \quad (j \in C),$$

where $|C| \geq 2$. By the affine-dependence observation in Lemma 3, for every $a \in C$, any affine hyperplane containing the profiles indexed by $C \setminus \{a\}$ also contains $\mathrm{prof}(a)$. So, no recent row realizes the trace $C \setminus \{a\}$ on C.

Since shattering J implies shattering C, all the traces $C \setminus \{a\}$ would have to be realized by distant rows. But distant row supports are nested prefixes, whereas the $|C| \geq 2$ sets $\{C \setminus \{a\} : a \in C\}$ are pairwise incomparable, and a chain cannot realize all of them. Therefore, this contradiction proves $\mathrm{VC}(\mathbf{M}^{(d)}) \leq d$.

Lower bound. We now use the prescribed assignment conditions from Section 3. These provide distinct distant positions $j_1, \ldots, j_D$ with $\mathrm{prof}(j_\ell) = e_\ell$ for $\ell \in [D]$, and a recent index $\tau \in [m]$ such that each witness hyperplane H_R, $\mathbf{R} \subseteq [D]$, occurs at recent query indices i_R^-, i_R^+ satisfying

$$i_R^- < \tau \leq i_R^+, \qquad H_{\mathrm{type}(i_R^-)} = H_{\mathrm{type}(i_R^+)} = H_R.$$

By the definition of these witness hyperplanes, $e_\ell \in H_R$ if and only if $\ell \in R$.

Consider the d column indices $J_* = \{j_1, \ldots, j_D, n+\tau\}$. Fix $A \subseteq J_*$ and set $\mathbf{R} = \{\ell : j_\ell \in A\}$. If $n + \tau \notin A$, use row $n + i_R^-$; if $n + \tau \in A$, use row $n + i_R^+$. In both cases, the incidence condition gives exactly the required trace on the distant landmarks. The recent causal triangle includes column $n+\tau$ precisely when the recent query index is at least τ, so the final coordinate also has its required label. Thus the selected row has trace exactly A on J_*.

Therefore, every subset of J_* is realized, giving $\mathrm{VC}(\mathbf{M}^{(d)}) \geq d$. Together with the upper bound, this proves $\mathrm{VC}(\mathbf{M}^{(d)}) = d$. □

VC dimension of learned multi-read routing. Fix a sequence and head, and let $D = d - 1 \geq 1$. With one profile per distant key and k positively weighted reads, each cross-block support is a union of k affine-hyperplane traces. Writing B_k for this binary cross-block mask and

$$\widehat{M}_k = \begin{pmatrix} L_n & 0 \\ B_k & L_m \end{pmatrix},$$

we have, for an absolute constant C,

$$\mathrm{VC}(\widehat{M}_k) \leq C(d-1)k \log_2(2k) + 2.$$

In particular, $\mathrm{VC}(\widehat{M}_4) \leq 25d - 24 = O(d)$.

Proof. Affine hyperplanes in $\mathbb{F}_q^D$ have VC dimension D. By Sauer-Shelah's lemma, the number of cross-block traces on any $s \geq D$ keys satisfies

$$\Pi_{B_k}(s) \leq \left(\sum_{r=0}^{D} \binom{s}{r} \right)^k \leq \left(\frac{es}{D} \right)^{Dk}.$$

Thus, shattering requires $2^s \leq (es/D)^{Dk}$. Setting $u = s/(Dk)$ gives $2^u \leq eku$, hence $u = O(\log_2(2k))$ and

$$\mathrm{VC}(B_k) = O(Dk \log_2(2k)).$$

For $k = 4$, setting $u = s/D$ instead yields $2^u \leq (eu)^4$, which implies $u < 25$. Consequently, $\mathrm{VC}(B_4) \leq 25D - 1$.

Finally, let $v = \mathrm{VC}(B_k)$. A shattered column set contains at most one recent column, since recent-column supports are nested prefixes. If it contains one, fixing that column to one forces recent rows to shatter all selected distant columns, giving size at most $v+1$. If all columns are distant, fixing the earliest to zero and the latest to one excludes every distant prefix row, so recent rows must shatter the remaining columns. Thus $\mathrm{VC}(\widehat{M}_k) \leq v + 2$, proving both claims. □

Dependence on the construction. The bounds above characterize our family under its prescribed field-size scaling. Other choices of q can preserve the same mask VC dimension while changing the number of profile cells, the number of tokens sharing each profile, and the computational cost. In particular, fixed q gives a fixed number of profiles as T grows. Our scaling instead lets the profile and type tables grow with sequence length.

E CHUNKING PROOFS

We now provide the proof for our result in Theorem 3.2.

Theorem 3.2 (Chunkwise SMat-attention)**.** *Fix $d \geq 1$, and let $\mathbf{M}^{(d)}$ be the mask of Section 3 under the prescribed sequence-length scaling. Assume the kernel has a nonnegative feature factorization of dimension r, and every normalizer $\bar{y}_{i,p}$ is positive. Then, for large T, Algorithm 1 computes the attention outputs $o_1, \ldots, o_T$, in*

$$O\big((T^{2-3/d} + T)\, rp \;+\; T\, c\,(r+p)\big) \tag{12}$$

work, excluding evaluation of the feature maps, using $O\big((T^{1-1/d} + T/c)rp + c^2\big)$ words of working memory. The backward pass has the same asymptotic cost.

Proof. **Forward Pass.** Before the final division, $\bar{y}_i$ is linear in the mask, and the two sub-masks of $\mathbf{M}^{(d)}$ have disjoint support. Thus, it suffices to evaluate each branch separately, add the outputs, and then normalize.

For the long-range branch, a recent token $n+i$ receives $\phi_{n+i}^\top \sum_{j \leq n} \mathbf{G}_{ij}\mathbf{Z}_j$. Substituting $\mathbf{G}_{ij} = \mathbf{C}_{\mathrm{type}(i),\mathrm{prof}(j)}$ and grouping the sum by profile,

$$\sum_{j \leq n} \mathbf{G}_{ij}\mathbf{Z}_j = \sum_{x \in \mathbb{F}_q^{d-1}} \mathbf{C}_{\mathrm{type}(i),x} \sum_{j \leq n:\, \mathrm{prof}(j)=x} \mathbf{Z}_j = \sum_x \mathbf{C}_{\mathrm{type}(i),x} F_x = U_{\mathrm{type}(i)},$$

which is the entry of the type table computed by the algorithm. The regrouping is valid because prof partitions $[n]$ into disjoint sets. Furthermore, because this long-range branch only applies to recent queries, the term evaluates to zero for all distant queries $\bar{y}_i^{\mathrm{lr}} = 0$ for $i \leq n$.

To be more explicit, each augmented state flattens into a row, so $\mathbf{Z} \in \mathbb{R}^{n \times rp}$. Recall the one-hot matrices of equation 8: $\mathbf{E} \in \{0,1\}^{n \times q^{d-1}}$ with $\mathbf{E}_{jx} = \mathbb{1}\{\mathrm{prof}(j) = x\}$, and $\mathbf{R} \in \{0,1\}^{m \times B}$ with $\mathbf{R}_{ih} = \mathbb{1}\{\mathrm{type}(i) = h\}$, so that $\mathbf{G} = \mathbf{R}\,\mathbf{C}\,\mathbf{E}^\top$. The long-range states of all recent queries are the rows of

$$\mathbf{G}\,\mathbf{Z} \;=\; \mathbf{R}\,\mathbf{C}\,\mathbf{E}^\top\mathbf{Z} \;=\; \mathbf{R}\Big(\,\mathbf{C}\Big(\;\underbrace{\underbrace{\mathbf{E}^\top\mathbf{Z}}_{\mathbf{F} \in \mathbb{R}^{q^{d-1} \times rp}}}_{U \in \mathbb{R}^{B \times rp}}\;\Big)\Big), \tag{15}$$
$$\in \mathbb{R}^{m \times rp}$$

and Algorithm 1 evaluates equation 15 from the right: $\mathbf{E}^\top Z$ is the pooling step, $\mathbf{C}(\cdot)$ the tabulation step, and the outer $\mathbf{R}$ the per-type read, which is a row gather whose inverse permutation is the scatter back into chronological order.

For the causal branch, fix $i \in I_b$. Because n is chunk-aligned, I_b lies entirely in $[n]$ or entirely in $(n, T]$. The incoming state collects the keys of all earlier chunks in that block and the within-chunk term collects $\{j \in I_b : j \leq i\}$, so their union is $\{j \leq i\}$ when $i \leq n$ and $\{n < j \leq i\}$ when $i > n$. These are the row supports of $\mathbf{L}_n$ and $\mathbf{L}_m$. Summing the branches and dividing once gives the output for $\mathbf{M}^{(d)}$.

For cost, the three factors of equation 15 correspond to three counts, each obtained from the last by collapsing one index, each worth a factor of $\Theta(T^{1/d})$. Note, $m, n = \Theta(T)$, $B = \Theta(q^{d-1})$ and $q = \Theta(T^{1/d})$.

From $\mathbf{G}$ itself, we have a hyperplane of $\mathbb{F}_q^{d-1}$ containing q^{d-2} of the q^{d-1} points (a $1/q$ of the entries are 1s), so a recent query is incident to $\Theta(n/q)$ distant keys and

$$\mathrm{nnz}(\mathbf{G}) = \Theta\big(mn/q\big) = \Theta\big(T^{2-1/d}\big), \tag{16}$$

Applying $\mathbf{R}$ collapses the queries – row i of $\mathbf{G}$ depends on i only through $\mathrm{type}(i)$ – so the m rows take only B distinct values, and the (type, token) incidences number

$$\Theta\big(Bn/q\big) = \Theta\big(nq^{d-2}\big) = \Theta(T^{2-2/d}),$$

which produces an additional saving of $m/B = \Theta(T^{1/d})$. Applying $E^\top$ collapses the keys in the same way: column j depends on j only through $\mathrm{prof}(j)$, so the n columns take only q^{d-1} distinct values, and thus the (type, profile) incidences are the nonzeros of $\mathbf{C}$,

$$\mathrm{nnz}(\mathbf{C}) = \Theta\big(Bq^{d-2}\big) = \Theta\big(q^{2d-3}\big) = \Theta\big(T^{2-3/d}\big), \tag{17}$$

a further saving of $n/q^{d-1} = \Theta(T^{1/d})$. Thus, tabulation performs one state addition of size rp per nonzero of $\mathbf{C}$, for $O(T^{2-3/d}rp)$.

The remaining steps are linear in T. Pooling reads each distant token once and adds it into one bucket; the per-type contractions cost rp per recent query, for $\sum_h m_h\, rp = m\, rp$ in total, independent of B; the chunk summaries and the scan touch each token and each of the T/c chunk boundaries a constant number of times. Together these are $O(Trp)$. Within a chunk, calculating the local attention scores costs $O(c^2 r)$ and applying them to the augmented values costs $O(c^2 p)$; summing over T/c chunks gives $O(Tc(r+p))$. Together, we have the stated bound.

Backward Pass. For the Long-range branch, let the three steps of equation 9 be

$$\begin{aligned}
\mathcal{P} : \mathbf{Z} \mapsto F, \qquad & F_x = \sum_{j \le n:\, \mathrm{prof}(j) = x} \mathbf{Z}_j, \\
\mathcal{C} : F \mapsto U, \qquad & U_h = \sum_x \mathbf{C}_{hx} F_x, \\
\mathcal{R}_\Phi : U \mapsto \bar{y}^{\mathrm{lr}}, \qquad & \bar{y}^{\mathrm{lr}}_{n+i} = \phi_{n+i}^\top U_{\mathrm{type}(i)},
\end{aligned}$$

so that $\bar{y}^{\mathrm{lr}} = \mathcal{R}_\Phi\, \mathcal{C}\, \mathcal{P}\, \mathbf{Z}$.

Let

$$\bar{g}_i := \frac{\partial \mathcal{L}}{\partial \bar{y}_i} \in \mathbb{R}^p$$

denote the gradient of the loss with respect to the augmented output. Pairing with $\bar{g}$ and moving one factor at a time across the inner product gives

$$\bar{U}_h = \sum_{i:\, \mathrm{type}(i) = h} \phi_{n+i}\, \bar{g}_{n+i}^\top, \quad \bar{F} = \mathbf{C}^\top \bar{U}, \quad \bar{\mathbf{Z}}_j = \bar{F}_{\mathrm{prof}(j)}, \quad \bar{\phi}_{n+i} = U_{\mathrm{type}(i)}\, \bar{g}_{n+i}. \tag{18}$$

The two grouped steps switch roles: the forward pass reduces over prof and broadcasts over type, while equation 18 reduces over type and broadcasts over prof. Each is one pass over the tokens it owns at $O(rp)$ per token, so both cost $O(Trp)$.

For the middle step, the incidence structure is biregular, i.e. every hyperplane of $\mathbb{F}_q^{d-1}$ contains q^{d-2} points, and also every point lies on $(q^{d-1}-1)/(q-1) = B/q$ hyperplanes (one per normalized direction a), since $x \in H_{a,b}$ forces $b = a^\top x$. Hence, $\mathrm{nnz}(\mathbf{C}^\top) = \mathrm{nnz}(\mathbf{C}) = \Theta(T^{2-3/d})$, and $\mathbf{C}^\top$ has constant column degree just as $\mathbf{C}$ has constant row degree. The tabulation step therefore applies to $\mathbf{C}^\top$ with the point-to-hyperplane table in place of the hyperplane-to-point one, giving $O(T^{2-3/d}rp)$ for $\bar{F}$, matching the forward calculation.

For the causal branch, fix a chunk I_b and stack its rows as $\Phi_b \in \mathbb{R}^{c\times r}$, $\Psi_b \in \mathbb{R}^{c\times r}$, $\bar{V}_b \in \mathbb{R}^{c\times p}$, and write $A^{(b)} := \mathrm{tril}(\Phi_b \Psi_b^\top)$ for the within-chunk score tile, so that equation 10 reads $Y_b = \Phi_b S_b^{\mathrm{in}} + A^{(b)} \bar{V}_b$. Differentiating this at fixed tril pattern,

$$\begin{aligned}
\bar{S}_b^{\mathrm{in}} &= \Phi_b^\top \bar{Y}_b, & \bar{\bar{V}}_b &= \big(A^{(b)}\big)^\top \bar{Y}_b, \\
\bar{\Phi}_b &= \bar{Y}_b \big(S_b^{\mathrm{in}}\big)^\top + \mathrm{tril}\big(\bar{Y}_b \bar{V}_b^\top\big)\Psi_b, & \bar{\Psi}_b &= \mathrm{tril}\big(\bar{Y}_b \bar{V}_b^\top\big)^\top \Phi_b.
\end{aligned} \tag{19}$$

Each is one $c \times c$ by $c \times r$ or $c \times p$ contraction, so $O(c^2(r+p))$ per chunk and $O(Tc(r+p))$ over the T/c chunks, the same as the forward tile. Note that $A^{(b)}$ and $\mathrm{tril}(\bar{Y}_b \bar{V}_b^\top)$ are rebuilt from $\Phi_b, \Psi_b, \bar{Y}_b, \bar{V}_b$ when the chunk is visited, so one c^2 tile is live at a time (same as in the forward pass).

For the scan, $S_b^{\mathrm{in}} = \sum_{a<b} D_a$ gives

$$\sum_b \big\langle \bar{S}_b^{\mathrm{in}}, \textstyle\sum_{a<b} D_a \big\rangle = \sum_a \Big\langle \sum_{b>a} \bar{S}_b^{\mathrm{in}},\ D_a \Big\rangle$$

so $\bar{D}_a = \sum_{b>a} \bar{S}^{\text{in}}_b$. In other words, the adjoint of an exclusive prefix scan is an exclusive suffix scan over the same chunks, run separately within $[n]$ and within $(n, T]$, at the same $O(T/c)$ state additions. Broadcasting $\bar{D}_b$ back to the tokens of I_b and adding the two contributions of equation 18 and equation 19 accumulates $\bar{\mathbf{Z}}_j$, from which $\bar{\psi}_j = \bar{\mathbf{Z}}_j \bar{v}_j$ and $\bar{\bar{v}}_j = \bar{\mathbf{Z}}_j^\top \psi_j$ follow pointwise in $O(rp)$ per token.

Finally since $o_i = \bar{y}_{i,1:d_v} / \bar{y}_{i,p}$ with $\bar{y}_{i,p} > 0$ by assumption, its Jacobian is row-local: with $\bar{o}_i := \frac{\partial \mathcal{L}}{\partial o_i}$,

$$\bar{g}_{i,1:d_v} = \frac{\bar{o}_i}{\bar{y}_{i,p}}, \qquad \bar{g}_{i,p} = -\frac{\bar{o}_i^\top \bar{y}_{i,1:d_v}}{\bar{y}_{i,p}^2} = -\frac{\bar{o}_i^\top o_i}{\bar{y}_{i,p}},$$

at $O(p)$ per token and $O(Tp)$ overall, which is dominated.

Every step of Algorithm 1 is therefore matched by an adjoint of the same shape and arithmetic. The backward pass runs in $O\big((T^{2-3/d} + T)rp + Tc(r+p)\big)$ work and $O\big((T^{1-1/d} + T/c)rp + c^2\big)$ words, the bounds of equation 12. □

Lemma 4 (Gated SMat)**.** *Under equation 13 with scalar per-token gates, Algorithm 1 computes the exact outputs within the complexity bound of equation 12, using $O(T)$ additional space.*

Lemmas 4– 5 concern additive kernel attention with optional scalar decay; they do not cover the delta-rule recurrence used in the GDN extension of Section 3.4.

Proof. We first compute $\lambda_1, \ldots, \lambda_T$ with one prefix sum, in $O(T)$ work and $O(T)$ space. Every factor used afterwards has the form $\exp(\lambda_i - \lambda_j)$ with $j \le i$. Since λ is non-increasing, each factor lies in $(0, 1]$, so we never form $\exp(\lambda_i)$ or $\exp(-\lambda_j)$ separately and no step can overflow.

Causal branch. Let $e(b)$ be the last index of chunk I_b. Each entry of the within-chunk tile becomes $(\phi_i^\top \psi_j) \exp(\lambda_i - \lambda_j)$, which is one entrywise product on a tile that is already materialized. The chunk summary becomes $D_b = \sum_{j \in I_b} \exp(\lambda_{e(b)} - \lambda_j) \mathbf{Z}_j$, which rescales each $\mathbf{Z}_j$ by a scalar prior to the sum. The carried state is read as $\exp(\lambda_i - \lambda_{e(b-1)})\, \phi_i^\top S^{\text{in}}_b$ and updated as $S^{\text{in}}_{b+1} = \exp(\lambda_{e(b)} - \lambda_{e(b-1)})\, S^{\text{in}}_b + D_b$, adding one scalar multiplication per chunk. The scan is otherwise unchanged.

Long-range branch. For a distant key $j \le n$ and a recent query $n+i$, the factor splits at the boundary,

$$\exp(\lambda_{n+i} - \lambda_j) = \exp(\lambda_{n+i} - \lambda_n) \cdot \exp(\lambda_n - \lambda_j),$$

into a query-side and a key-side scalar, each at $(0, 1]$. The key-side scalar is folded into $\mathbf{Z}_j$ before pooling, and the query-side scalar multiplies the query's read from the type table. Neither modifies $\mathbf{C}$, the profile table or the type table, so the long-range branch is the ungated computation applied to rescaled inputs.

The added work is $O(T)$ exponentials, $O(Tc)$ for the tile products and $O(Tr)$ to rescale features, all dominated by terms already in equation 12. The added space is the $O(T)$ values of λ. □

Note, the same factorization gives streaming decoding. Let $a_t = \exp(\lambda_t - \lambda_{t-1}) \in (0, 1]$. At the boundary, cache the type-major states

$$U_h^{(n)} = \sum_{j \le n} C_{h,\text{prof}(j)}\, \exp(\lambda_n - \lambda_j)\, \mathbf{Z}_j,$$

which is the ungated cache with each $\mathbf{Z}_j$ rescaled. For $t > n$, maintain the recent state $S_t = a_t S_{t-1} + \mathbf{Z}_t$ and the scalar $g_t = a_t g_{t-1}$, starting from $S_n = 0$ and $g_n = 1$. By induction, $S_t = \sum_{n < j \le t} \exp(\lambda_t - \lambda_j) \mathbf{Z}_j$ and $g_t = \exp(\lambda_t - \lambda_n)$, so the augmented output is

$$\bar{y}_t = \phi_t^\top \Big[g_t\, U^{(n)}_{\text{type}(t-n)} + S_t \Big].$$

Each token costs the same as as the ungated computation plus two scalar multiplications.

F Decoding and Horizon-Free Decoding

We first restate and prove Theorem 3.3.

Theorem 3.3 (Streaming decoding)**.** *Assume the hypotheses of Theorem 3.2, and suppose the total sequence length T is fixed in advance so that the distant/recent boundary n is known. Once Algorithm 1 has been run over the distant block, each subsequent token can be decoded in $O(rp)$ time, independent of T, from a cache of $O(T^{1-1/d}rp)$ words.*

Proof. The type table $\{U_h\}_{h<B}$ depends only on the distant tokens. Since every distant position precedes every recent one, it is fixed once the distant block has been consumed, and hence, the profile states $\{F_x\}$ can be discarded. By the decomposition in the proof of Theorem 3.2, the augmented output at recent position $n+i$ is

$$\bar{y}_{n+i} = \phi_{n+i}^{\top}\Big(U_{\mathrm{type}(i)} + \sum_{n<j\le n+i} \mathbf{Z}_j\Big), \tag{20}$$

and the second term is a single running state S maintained by the in-place update $S \leftarrow S+\mathbf{Z}_{n+i}$. The index $\mathrm{type}(i) = i \bmod B$ is one arithmetic operation. Forming $\mathbf{Z}_{n+i}$, updating S, and contracting $\phi_{n+i}^{\top}(U_{\mathrm{type}(i)} + S)$ each cost $O(rp)$. The retained state is the B tabulated states together with S. For the gated variant from Equation 13, the running state update becomes $S \leftarrow a_{n+i}S + \mathbf{Z}_{n+i}$ and the read of $U_{\mathrm{type}(i)}$ carries the scalar $\exp(\lambda_{n+i} - \lambda_n)$, maintained by one addition per token. □

The streaming-decoding theorem (Theorem 3.3) assumes that the total length T of the token sequence is known in advance, since the distant/recent boundary n and the field size q are both functions of T and together determine the structure of the mask. Hence, if more tokens were to be added beyond T, the layer degrades to linear attention. In this section we replace that assumption with a fixed training horizon $T_{\max}$ and a recent window that advances in steps of c_{dc}.

Definition 1 (Stepped-window mask)**.** *Fix a training horizon $T_{\max}$, a block length c_{dc} dividing $T_{\max}$ and a multiple of the chunk width c, and the geometry $(q, \mathbf{C}, B)$ from the SMat construction of Section 3 for $T = T_{\max}$. For $t \ge 1$ let $b(t) = \lfloor (t-1)/c_{dc} \rfloor$ be the block of position t and $n(t) = \max\{0,\ (b(t)-1)c_{dc}\}$ be its number of distant positions. For any T, including $T > T_{\max}$,*

$$\mathbf{M}^{(\mathrm{step})}_{tj} = \begin{cases} 1 & n(t) < j \le t \quad (\textit{recent window}), \\ \mathbf{C}_{\mathrm{type}(t),\,\mathrm{prof}(j)} & j \le n(t) \quad (\textit{distant, through } \mathbf{G}), \\ 0 & j > t. \end{cases}$$

In the block form of Section 3 the boundary $n = c\lfloor T/2c \rfloor$ is a constant determined by T. Here, the boundary is the function $n(\cdot)$, of which the constant n is the special case $n(t) \equiv n$. Since $n(t)$ depends only on t, the mask for a length-t sequence is the leading $t \times t$ block of the mask for any longer sequence. Thus, training at $T_{\max}$ and decoding at any length uses one mask.

In Algorithm 3, each fresh recurrent state is initialized to zero, with update $R \leftarrow R + Z_t$ and readout $\mathrm{read}(R, \phi_t) = \phi_t^{\top} R$. For the scalar-gated variant, the update becomes $R \leftarrow a_t R + Z_t$.

Algorithm 3 Stepped-window decoding for one attention head

Require: geometry $(q, \mathbf{C}, B)$ fixed from $T_{\max}$; block length c_{dc}; maps $\mathrm{prof}, \mathrm{type}$
1: $U \leftarrow 0$; $\Delta_{\mathrm{old}}, \Delta_{\mathrm{cur}} \leftarrow 0$; $R_{\mathrm{old}}, R_{\mathrm{cur}} \leftarrow$ fresh recurrent states
2: **for** $t = 1, 2, \ldots$ **do**
3: **if** $t > 1$ **and** $b(t) \ne b(t-1)$ **then**
4: **if** $b(t) \ge 2$ **then** $U \leftarrow U + \mathbf{C}\,\Delta_{\mathrm{old}}$
5: $\Delta_{\mathrm{old}} \leftarrow \Delta_{\mathrm{cur}}$; $\Delta_{\mathrm{cur}} \leftarrow 0$; $R_{\mathrm{old}} \leftarrow R_{\mathrm{cur}}$; $R_{\mathrm{cur}} \leftarrow$ fresh
6: Compute ϕ_t and $\mathbf{Z}_t = \psi_t \bar{v}_t^{\top}$; update $R_{\mathrm{old}}, R_{\mathrm{cur}}$; $\Delta_{\mathrm{cur}}[\mathrm{prof}(t)]$ += $\mathbf{Z}_t$.
7: $\bar{y}_t \leftarrow \phi_t^{\top} U_{\mathrm{type}(t)} + \mathrm{read}(R_{\mathrm{old}}, \phi_t)$; $o_t \leftarrow \bar{y}_{t,1:d_v} / \bar{y}_{t,p}$.

Lemma 5 (Horizon-free streaming decoding)**.** *Assume the hypotheses of Theorem 3.2, with the geometry fixed from $T_{\max}$ and $c_{dc} = T_{\max}/4$. Then Algorithm 3 computes the outputs of the stepped-window mask of Definition 1, for every length T at $O\big((1+T_{\max}^{1-3/d})rp\big)$ work per token from a cache of $O(T_{\max}^{1-1/d})$ states of size $r \times p$.*

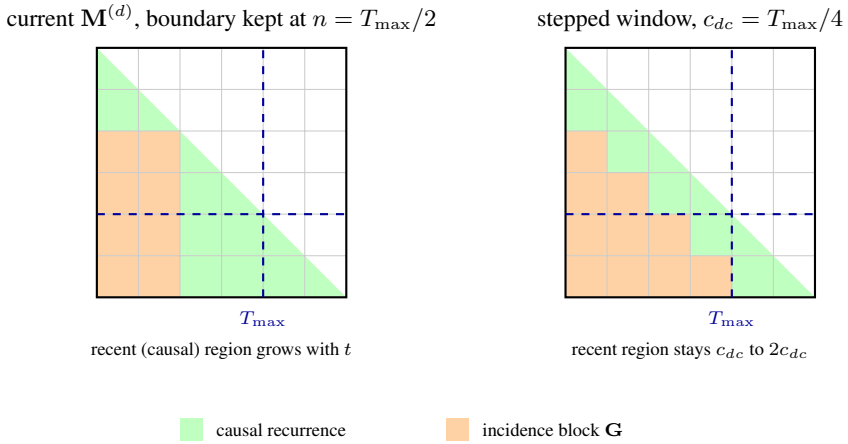

Figure 6: Both masks drawn at the same length $T = 1.5\,T_{\max}$, in blocks of $c_{dc} = T_{\max}/4$ tokens. **Left:** the fixed-T mask once its boundary is kept at $n = T_{\max}/2$. The incidence block $\mathbf{G}$ stops growing at two blocks and everything after the boundary falls to the recurrence. **Right:** the stepped window, whose recent region stays between c_{dc} and $2c_{dc}$ tokens at every length while $\mathbf{G}$ keeps absorbing the older blocks.

Proof. Index blocks so that block β occupies positions $\beta c_{dc} + 1, \ldots, (\beta + 1)c_{dc}$, and write $\beta(t) = b(t)$. For $0 \le n \le T$ let

$$F_x^{(n)} := \sum_{j \le n,\ \mathrm{prof}(j) = x} \mathbf{Z}_j, \qquad x \in \mathbb{F}_q^{d-1},$$

be the profile table of the first n positions, viewed as a $q^{d-1} \times rp$ matrix. Since $n(t) = (\beta(t) - 1)c_{dc}$ for $\beta(t) \ge 1$, the distant region of a query in block β is the union of blocks $0, \ldots, \beta - 2$.

We claim that for every $t \ge 1$, by line 7 in the algorithm,

(i) $U = \mathbf{C}F^{(n(t))}$;

(ii) Δ_{old} holds the per-profile sums of block $\beta(t) - 1$, and Δ_{cur} those of block $\beta(t)$ restricted to positions $\le t$;

(iii) R_{old} is the state of the recurrence initialized at position $n(t) + 1$ and advanced through t.

We prove this by induction on t. For $t = 1$ we have $\beta(1) = 0$ and $n(1) = 0$, the conditional does not fire, and $U = 0 = \mathbf{C}F^{(0)}$, giving (i)–(iii). Assume the invariant at $t - 1$ and consider t.

If $\beta(t) = \beta(t - 1)$ then $n(t) = n(t - 1)$ and lines 4–5 are skipped, so U is unchanged and (i) persists; line 6 adds $\mathbf{Z}_t$ to $\Delta_{\mathrm{cur}}[\mathrm{prof}(t)]$ and advances both recurrences, which preserves (ii) and (iii).

If $\beta(t) \ne \beta(t - 1)$ then $t = \beta c_{dc} + 1$ and $t - 1$ is the last position of block $\beta - 1$. For $\beta \le 1$ we have $n(t) = 0$, the guard on line 4 suppresses the update, and (i) holds with $U = 0$. For $\beta \ge 2$, the inductive hypothesis at $t - 1$ gives $U = \mathbf{C}F^{(n(t-1))} = \mathbf{C}F^{((\beta-2)c_{dc})}$ and, by (ii), Δ_{old} equal to the per-profile sums of block $\beta - 2$. Those sums are precisely $F^{((\beta-1)c_{dc})} - F^{((\beta-2)c_{dc})}$, so line 4 yields

$$U \;=\; \mathbf{C}F^{((\beta-2)c_{dc})} + \mathbf{C}\Big(F^{((\beta-1)c_{dc})} - F^{((\beta-2)c_{dc})}\Big) \;=\; \mathbf{C}F^{((\beta-1)c_{dc})} \;=\; \mathbf{C}F^{(n(t))},$$

which shows (i). Line 5 then sets Δ_{old} to the sums of block $\beta - 1$ and clears Δ_{cur}, and line 6 adds $\mathbf{Z}_t$, giving (ii); and it reassigns $R_{\mathrm{old}} \leftarrow R_{\mathrm{cur}}$, where R_{cur} was initialized at the first position of block $\beta - 1$, namely $(\beta - 1)c_{dc} + 1 = n(t) + 1$, giving (iii).

Exactness. The augmented output is linear in the mask row, so by Definition 1,

$$\bar{y}_t = \sum_j \mathbf{M}^{(\mathrm{step})}_{tj}\, \phi_t^\top \mathbf{Z}_j = \phi_t^\top \Big(\underbrace{\sum_{j \le n(t)} \mathbf{C}_{\mathrm{type}(t),\mathrm{prof}(j)} \mathbf{Z}_j}_{\text{distant}} + \underbrace{\sum_{n(t)<j\le t} \mathbf{Z}_j}_{\text{recent}} \Big). \tag{21}$$

Grouping the distant keys by profile gives $\sum_{j\le n} \mathbf{C}_{h,\mathrm{prof}(j)} \mathbf{Z}_j = (\mathbf{C} F^{(n)})_h$, so by invariant (i) the first term of equation 21 equals $\phi_t^\top U_{\mathrm{type}(t)}$. The recurrence is linear and its state is initialized to zero at $n(t)+1$, so by invariant (iii) it holds $\sum_{n(t)<j\le t} \mathbf{Z}_j$ and the second term equals $\mathrm{read}(R_{\mathrm{old}}, \phi_t)$. Both are accumulated into $\bar{y}_t$ before the single division on line 7, so the normalized output is exact.

Gating. For the gated mask $\widetilde{\mathbf{M}}_{tj} = \mathbf{M}^{(\mathrm{step})}_{tj} \exp(\lambda_t - \lambda_j)$, replace $F^{(n)}$ by the table referred to the last distant position, $\widetilde{F}^{(n)}_x := \sum_{j\le n,\ \mathrm{prof}(j)=x} \exp(\lambda_n - \lambda_j)\mathbf{Z}_j$, and read U as $\exp(\lambda_t - \lambda_{n(t)})\, \phi_t^\top U_{\mathrm{type}(t)}$. Writing $N = (\beta-2)c_{dc}$ and $N' = (\beta-1)c_{dc}$, the two tables satisfy

$$\widetilde{F}^{(N')} = \exp(\lambda_{N'} - \lambda_N)\, \widetilde{F}^{(N)} + \Big(\text{per-profile sums of block } \beta-2 \text{ referred to } N'\Big),$$

so invariant (i) is restored with line 4 replaced by $U \leftarrow \exp(\lambda_{N'} - \lambda_N) U + \mathbf{C}\, \Delta_{\mathrm{old}}$, one scalar multiply per block. The induction and the exactness argument are otherwise unchanged.

Cost. Per token, forming ϕ_t and $\mathbf{Z}_t$, the two recurrent updates, the scatter-add into Δ_{cur}, and the read and normalization on line 7 are each $O(rp)$. The only other work is line 4, one sparse product costing $\mathrm{nnz}(\mathbf{C})\, rp$ scalar operations once every c_{dc} tokens. As written, line 4 performs the sparse product at the first token of each block, so the algorithm meets the bound below in amortized form. For a worst-case bound, the next type table $U + \mathbf{C}\, \Delta_{\mathrm{old}}$ is built in a second buffer during the preceding block: the block it absorbs is complete for all c_{dc} steps of that block, so copying U ($B\, rp$ operations) and the sparse product ($\mathrm{nnz}(\mathbf{C})\, rp$) can be spread over those steps, and the buffers are swapped only once the new table is complete. Since $B \le \mathrm{nnz}(\mathbf{C})$, this adds $O\big((1 + \mathrm{nnz}(\mathbf{C})/c_{dc})\, rp\big)$ work to each token. In the regular construction, the number of normalized directions is $|\mathcal{A}| = (q^{d-1}-1)/(q-1) = \Theta(q^{d-2})$ and each profile lies on one hyperplane per direction, so

$$\mathrm{nnz}(\mathbf{C}) = |\mathcal{A}|\, q^{d-1} = \Theta(q^{2d-3}).$$

With $T_{\max} = \Theta(q^d)$ this is $\mathrm{nnz}(\mathbf{C}) = \Theta(T_{\max}^{2-3/d})$, and since $c_{dc} = \Theta(T_{\max})$ the per-token share is $\Theta(T_{\max}^{1-3/d})$, giving $O\big((1 + T_{\max}^{1-3/d}) rp\big)$, which is $O(rp)$ exactly when $d \le 3$.

Cache. The retained state is the type table U (B states), the two profile deltas (q^{d-1} states each), the two recurrent states, and the second type table used above. Since $B = \Theta(q^{d-1}) = \Theta(T_{\max}^{1-1/d})$, the total is $O(T_{\max}^{1-1/d})$ states of size $r \times p$. □

Other dynamic boundary candidates. Write $n(t)$ for the boundary each schedule places at query t, and T_{train} for the training length. We compare six boundary schedules.

1. *No long-range branch* sets $n(t) = 0$, so $\mathbf{G}$ is never used and the layer is the ordinary recurrence, the $d = 1$ SMat member.
2. *Boundary kept, current* is the mask of the main text with $n(t) = T_{\mathrm{train}}/2$ held where prefill placed it, which is what the fixed-horizon decoder does when it is run past its horizon.
3. *Rebuilt, current* is the same mask recomputed at the evaluation length, $n(t) = T/2$. It is computationally expensive and needs T in advance.
4. *Doubling* places the boundary at $n(t) = 2^{\lfloor \log_2 t \rfloor - 1}$, so it doubles at each power of two.
5. The two *window* arms are definition 1 with $c_{dc} = T_{\mathrm{train}}/4$ and $c_{dc} = T_{\mathrm{train}}/8$.

All arms share architecture, data, optimizer and step budget. We demonstrate capabilities with the $d = 2$ SMat mask: a Mamba-2-style decayed linear recurrence reset at $n(t)$, plus the incidence branch read by a fixed random hash of token identifiers of Section 3. Each result is the mean over three seeds, with the sample standard deviation as a subscript.

Schedule	1024	2048	4096	8192
No long-range branch	13.3 (10.0)	13.5 (10.6)	12.8 (9.6)	11.7 (9.4)
Boundary kept, current	91.5 (3.7)	78.4 (5.3)	63.6 (9.2)	51.9 (11.2)
Rebuilt, current	91.6 (4.0)	87.2 (4.7)	83.7 (6.1)	81.8 (6.8)
Doubling	78.8 (4.0)	68.7 (4.2)	62.6 (1.8)	53.9 (1.5)
Window, $c_{dc} = T_{\max}/8$	93.8 (4.9)	98.3 (1.4)	95.6 (6.8)	90.1 (13.9)
Window, $c_{dc} = T_{\max}/4$	82.2 (2.3)	92.9 (1.0)	95.9 (0.5)	97.6 (1.1)

Table 5: MQAR accuracy (%) by evaluation length, *uniform* layout, models trained at length 1024. The quarter-length window is the only schedule whose accuracy rises with the evaluation length, and the only one whose spread does not. Knowing T in advance recovers part but not all of the fixed boundary's loss: the rebuilt oracle ends at 81.8 against 51.9 for the kept boundary. Mean (std) over 3 seeds.

Schedule	$D = 0$	$D = 256$	$D = 1024$	$D = 3072$	$D = 7678$
No long-range branch	8.7 (6.1)	7.9 (5.7)	8.3 (6.3)	8.0 (6.4)	7.0 (5.8)
Boundary kept, current	35.4 (14.9)	34.7 (15.7)	34.1 (15.1)	33.6 (14.7)	99.6 (0.3)
Rebuilt, current	36.1 (18.4)	34.9 (18.0)	35.9 (18.9)	34.6 (18.8)	99.7 (0.3)
Doubling	22.4 (3.8)	23.1 (2.4)	23.0 (3.2)	22.1 (3.1)	85.6 (7.5)
Window, $c_{dc} = T_{\max}/8$	73.4 (20.2)	81.9 (23.0)	81.7 (22.5)	81.9 (22.9)	83.9 (20.1)
Window, $c_{dc} = T_{\max}/4$	27.1 (3.0)	94.9 (3.2)	94.3 (3.3)	94.7 (3.0)	97.5 (2.6)
Fraction of query rows whose pair region is routed through $\mathbf{G}$					
Fixed boundary / doubling	0.00	0.00	0.00	0.00	1.00
Window, $c_{dc} = T_{\max}/4$	0.00	1.00	1.00	1.00	1.00

Table 6: MQAR accuracy (%) by gap D between the pair region and the query region, *placed* layout at evaluation length 8192. The lower block reports how often the pairs actually reach the long-range branch. At $D = 7678$ the pairs precede every boundary and the fixed schedules are the strongest arms. For $256 \leq D \leq 3072$ the pairs land inside the fixed boundary's recent region, which it can serve only from the fixed-size recurrence, and its accuracy falls to about 34% while the quarter-length window holds about 95%. At $D = 0$, the quarter-length window obtains 27.1%, below the two fixed-boundary variants (35.4% and 36.1%). The eighth-length window instead obtains 73.4%, with substantial seed variation. Performance near the boundary therefore depends on the particular window schedule. Mean (std) over 3 seeds.

Task and evaluation layouts. Each MQAR sequence contains 64 key–value pairs over a noise vocabulary. Every key occurs twice, once beside its value and once as a query, and the model must emit that value at the query. During training the pair and query positions are drawn uniformly over the sequence. At evaluation we use two layouts. Under *uniform* the positions are drawn as in training, so the result measures length generalization alone. Under *placed* all pairs are confined to a 256-token region and all queries to a 256-token region at the very end of the sequence, separated by a gap of D tokens; D controls how far back a query must reach, with large D putting the pairs near the start of the context. The placed layout moves the stored content across each schedule's boundary while holding everything else fixed.

Length generalization. Table 5 evaluates each schedule out to eight times its training length under the training distribution.

Recall relative to the boundary. Table 6 fixes the evaluation length at 8192 and sweeps the gap D. The final column, $D = 7678$, places the pairs at the very start of the sequence, where every schedule routes them through $\mathbf{G}$; it is a control rather than a hard case.

Language modelling. Table 7 reports byte-level PG-19, trained at length 2048 and evaluated on 16384-byte windows.

Schedule	$[0, 1024)$	$[2048, 4096)$	$[4096, 8192)$	$[8192, 16384)$
No long-range branch	1.686 (0.006)	1.675 (0.007)	1.679 (0.008)	1.670 (0.007)
Boundary kept, current	1.634 (0.005)	1.588 (0.005)	1.598 (0.004)	1.595 (0.005)
Rebuilt, current	1.634 (0.005)	1.614 (0.008)	1.619 (0.006)	1.601 (0.004)
Doubling	1.646 (0.008)	1.605 (0.008)	1.610 (0.007)	1.610 (0.008)
Window, $c_{dc} = T_{\max}/8$	1.608 (0.003)	1.546 (0.003)	1.561 (0.004)	1.559 (0.004)
Window, $c_{dc} = T_{\max}/4$	1.639 (0.008)	1.584 (0.012)	1.596 (0.010)	1.595 (0.014)

Table 7: PG-19 bits per byte by position within the evaluation window, trained at length 2048 and evaluated at 16384; lower is better. Every schedule improves past the training length rather than degrading, and the quarter-length window is similar to the fixed boundary. Mean (std) over 3 seeds.

G Training Details and additional results

All runs use an Adam-family optimizer, gradient-norm clipping at 1.0, and a single GPU per run. Learning-rate schedules are linear warmup followed by cosine decay to $0.1\times$ the peak, except on subset routing and multi-key subset recall, which train at a constant rate. Weight decay is applied to matrix parameters only; biases, norms, and the recurrent parameters $A_{\log}$, Δ_{bias}, and D are excluded. Every arm within a table shares its data order, schedule, and precision.

	Routing	**MQAR**	**MKAR**	**JCKR**	**PG-19**
Data					
Sequence length	1024	64–256	1024	64–3076	16384 / 32768
Vocabulary	–	8192	–	563	50257 (GPT-2)
Task size	$k \leq 6$	4–64 pairs	8 pairs	4–512 records	–
Model					
Layers	1	2	1	2	8
Width d_{model}	64	16/32/64	256	64	384
Head dim.	–	16	64	16	64
State dim.	–	16	64	16	64
Short conv. width	–	3	3	4	4
Optimization					
Optimizer	Adam	AdamW	AdamW	AdamW	AdamW
Peak LR	3e-3	1e-2	3e-4	3e-3	6e-4
Weight decay	0	0.1	0.1	0.1	0.1
Batch size	64	256	32	256	2 / 1
Budget	1500 steps	32 epochs	12000 steps	32 epochs	9155 steps
Warmup	0	–	0	0	300
Precision	fp32	bf16	bf16	bf16	bf16
Seeds	3	3	3	3	1

Table 8: Training configuration for each experiment. Routing trains with the mask fixed and optimizes a mean-squared-error readout; MKAR uses binary cross-entropy on one sigmoid logit per pair (multi-label), and MQAR, JCKR and PG-19 use categorical (softmax) cross-entropy. PG-19 batch sizes are per context length, chosen so that every step sees 32,768 tokens, giving 300M training tokens per arm.

G.1 Extension Details

We describe one attention head and suppress the head index. Let h_t denote the layer input at position t, which depends only on tokens at positions $1, \ldots, t$. The distant/recent boundary n and the finite geometry are fixed independently of token content.

Causal write addresses. The write-side representation is a short causal convolution,

$$c_t = \sum_{\ell=0}^{L-1} K_\ell h_{t-\ell},$$

with zero padding before the sequence begins; our implementation uses a depthwise convolution with $L = 4$. This allows a write to be addressed using nearby preceding content, such as a key preceding its value. Write addresses are computed when tokens are processed and are not revised using later queries. Let $D = d - 1$. For each coordinate $k \in [D]$, we compute

$$z_{t,k} = \gamma_k \left\langle \frac{w_k}{\|w_k\|_2}, \mathrm{LN}(c_t) \right\rangle + b_k,$$

and

$$s_{t,k} = \mathrm{clip}\big(q\Phi(z_{t,k}),\, 0,\, q - \varepsilon\big),$$

where LN normalizes each token independently, Φ is the standard normal CDF, and $\varepsilon > 0$ is a small numerical constant. The discrete hash is

$$x_\theta(c_t) = \big(\lfloor s_{t,1} \rfloor, \ldots, \lfloor s_{t,D} \rfloor\big) \in \mathbb{F}_q^D.$$

Thus, $\mathrm{prof}(j) = x_\theta(c_j)$ determines the profile memory updated by distant token j. The projection parameters (w_k, γ_k, b_k) are learned jointly with the backbone; the geometric incidence matrix C remains fixed.

Learning the discrete hash. The floor operation has no useful ordinary derivative. Therefore, following Jiang et al. (2018), we use a straight-through estimator: the forward pass uses a discrete address, while the backward pass uses a surrogate based on interpolation between neighboring bins. Write

$$a_{t,k} = \lfloor s_{t,k} \rfloor, \qquad f_{t,k} = s_{t,k} - a_{t,k}.$$

The corresponding interpolation assigns weights $1 - f_{t,k}$ and $f_{t,k}$ to bins $a_{t,k}$ and $\min(a_{t,k}+1, q-1)$, respectively. In the sparse four-read implementation, we retain only the selected write route. Its straight-through weight is

$$\omega_t = \prod_{k=1}^{D} \left[1 + (1 - f_{t,k}) - \mathrm{sg}(1 - f_{t,k})\right],$$

where sg denotes stop-gradient. Numerically, $\omega_t = 1$, so each token writes to exactly one profile during both training and inference. During backpropagation, the write weight supplies a surrogate gradient to the hash parameters while the discrete address is held fixed. This is a one-sided, biased straight-through estimator, rather than differentiation through the discrete bin index.

We also encourage balanced coordinate occupancy. Let $\bar{p}_{k,b}$ be the average soft interpolation mass assigned to bin b of coordinate k over distant write positions. The auxiliary loss is

$$\mathcal{L}_{\mathrm{bal}} = \frac{1}{D} \sum_{k=1}^{D} \sum_{b=0}^{q-1} \bar{p}_{k,b} \log\big(q\bar{p}_{k,b}\big), \qquad \mathcal{L} = \mathcal{L}_{\mathrm{task}} + \beta_{\mathrm{bal}} \mathcal{L}_{\mathrm{bal}}.$$

This penalizes concentration in individual coordinate bins; it does not require uniform occupancy of all joint profiles.

Query routing and causality. After processing the distant block, profile memories are aggregated into type summaries,

$$U_h = \sum_x C_{hx} F_x.$$

For the single-hyperplane construction, a recent query at $t = n + i$ hashes its causal representation to a point u_t and chooses a nonzero normalized direction a_t using that same representation. Its type is $H_{a_t, a_t^\top u_t}$, with arithmetic over $\mathbb{F}_q$. Consequently, its long-range support is

$$G_{ij} = \mathbf{1}\left\{a_{n+i}^\top \mathrm{prof}(j) = a_{n+i}^\top u_{n+i}\right\}, \qquad j \le n.$$

This guarantees visibility for matching hash points. Repeated token identities alone need not produce matching points when the hash inputs include contextual information.

The four-read variant instead uses an independent learned scorer $\ell_{i,h}$ computed from h_{n+i}. Let $\mathcal{S}_i = \mathrm{Top4}_h(\ell_{i,h})$. Its read weights are

$$R_{ih} = \begin{cases} \dfrac{\exp(\ell_{i,h})}{\sum_{g \in \mathcal{S}_i} \exp(\ell_{i,g})}, & h \in \mathcal{S}_i, \\ 0, & \text{otherwise.} \end{cases}$$

The selected scores receive ordinary gradients through this softmax; we do not differentiate through the top-four indices. Unlike the single-hyperplane read, this selector need not choose a hyperplane through the query's own hash point.

Both variants are causal: every summary contains only positions $j \le n < n + i$, and the query's routing decision depends only on its causal representation. Combined with the causal local branch, the output at position t depends only on tokens at positions at most t.

Memory updates. The memory updates are adapted from the recurrent dynamics of the corresponding backbone, with new payloads routed to learned profile addresses.

For Mamba-2 extensions, we use an additive memory update: $F_x^{(j)} = a_j^G F_x^{(j-1)} + \mathbf{1}\{x_j = x\}\omega_j k_j v_j^\top$, where x_j is the learned write profile and ω_j is an independent sigmoid write gate. We reuse the backbone's input-dependent retention factor $a_j^G = \exp(-\exp(\theta_A)\Delta_j)$, where θ_A is a learned parameter and $\Delta_j > 0$ is the input-dependent time step.

For GDN extensions, we use boundary-transported additive memory, incorporating the backbone's delta-rule transitions. Let k_t, q_t be the normalized backbone keys and queries, v_t the values, β_t the write gate, and a_t the scalar decay. Define $A_t = a_t(I - \beta_t k_t k_t^\top)$. For distant token $j \le n$ with learned profile x_j, the boundary-transported key is

$$\bar{k}_j = A_n \cdots A_{j+1}\beta_j k_j, \qquad F_x = \sum_{j \le n:\, x_j = x} \bar{k}_j v_j^\top,$$

where an empty product is the identity. Equivalently, starting from zero, $F_x^{(j)} = A_j F_x^{(j-1)} + \mathbf{1}\{x_j = x\}\beta_j k_j v_j^\top$. Thus every profile undergoes the shared transition A_j, while only the selected profile receives the new payload. The implementation computes transported keys and pools them additively, rather than explicitly updating every profile.

After forming $U_b = \sum_x C_{bx} F_x$, a recent token $t > n$ uses

$$\bar{q}_t = r^{-1/2}(A_t \cdots A_{n+1})^\top q_t, \qquad z_t = z_t^{\mathrm{local}} + \lambda_t \bar{q}_t^\top \sum_b R_{tb} U_b.$$

Here R_{tb} contains the four selected softmax read weights and λ_t is a learned sigmoid gate.

G.2 Experiments

Exact-support match. Both subset routing and Multi-key subset recall tasks are scored by whether the model retrieves exactly the requested items. A query counts as correct only if every requested item is present in the output and every unrequested item is absent. In subset routing the output has one channel per marked position, and a channel counts as present when its magnitude exceeds 1% of the largest payload in that example. In multi-key subset recall the output has one raw logit z_i per key–payload pair $i \in \{1, \ldots, 8\}$, with no sigmoid or softmax applied before thresholding. A pair counts as present when $z_i > 0.5$, i.e. when its sigmoid probability exceeds $\sigma(0.5) \approx 0.62$, and the query is correct when $\{i : z_i > 0.5\}$ equals the requested set. Training uses the matching per-pair binary cross-entropy, $-\frac{1}{8}\sum_i \left[y_i \log \sigma(z_i) + (1 - y_i)\log(1 - \sigma(z_i))\right]$ averaged over answer rows, with a k-hot target y. Each evaluation draws 2,048 fresh queries (8 batches of 32 sequences, 8 answer rows each).

Subset routing. Payloads and requested subsets are resampled every batch, giving roughly 10^5 examples per run against at most 2^6 distinct requests. Reported values are the mean of three seeds.

Model	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$
Softmax	100.00 (0.00)	52.21 (2.09)	28.78 (0.98)	17.97 (3.47)	7.81 (1.28)	5.27 (0.90)
Mamba-2	100.00 (0.00)	63.09 (9.95)	32.23 (6.60)	19.21 (4.80)	9.90 (1.64)	5.60 (0.81)
DeltaNet	99.93 (0.11)	76.69 (1.85)	46.55 (2.35)	30.21 (4.35)	15.76 (0.45)	9.83 (1.18)
Gated DeltaNet	100.00 (0.00)	66.80 (13.44)	39.78 (5.18)	24.22 (6.80)	16.93 (0.30)	8.27 (2.14)
Log-Linear	99.93 (0.11)	76.69 (1.71)	46.55 (4.30)	28.71 (5.40)	16.41 (1.55)	9.77 (0.59)
SMat ($d=1$)	100.00 (0.00)	77.21 (1.75)	48.31 (4.56)	29.88 (3.65)	14.65 (1.35)	9.96 (1.03)
SMat ($d=2$)	99.93 (0.11)	99.48 (0.23)	82.49 (1.77)	52.93 (1.17)	33.07 (1.44)	17.12 (3.00)
SMat ($d=3$)	99.93 (0.11)	99.48 (0.23)	98.18 (0.56)	89.45 (3.07)	70.18 (2.74)	43.10 (3.85)
SMat ($d=4$)	99.93 (0.11)	99.54 (0.11)	98.18 (0.56)	97.98 (0.98)	93.68 (1.08)	78.71 (2.25)

Table 9: Exact routing-pattern match (%) on subset routing, by the number k of marked positions. $T = 1024$, one layer, 1500 steps, batch 64, learning rate 0.003; payloads are resampled each batch and evaluation uses fresh ones. Mean (std) over 3 seeds. SMat performs better at higher k as d increases.

The causal mask has VC dimension one because its row supports form a nested family, and softmax attention on the mask can assign substantially different weights to keys within an allowed prefix. The baseline results in Table 9 therefore describe performance under the evaluated architecture, optimization, and scoring protocol.

Verification of end-to-end cost. Figure 3 times the prefill of a single SMat layer. Here we measure the full cost of the models that produce these subset routing results. Namely, we analyze training steps (forward pass, backward pass and optimizer update), prefill, and per-token decoding, together with peak memory and decoding cache. All measurements use one NVIDIA H200 GPU with PyTorch 2.11 and Triton 3.7.1, random weights and random inputs (step time depends on shapes, not on trained values). We report the median of at least ten timed steps after warm-up.

We note some critical testing implementation details. The SMat prefill is reported with custom Triton kernels, but SMat training was done through its Pytorch path (we did not develop the kernels for backwards pass). Softmax attention uses PyTorch SDPA, which in fp32 runs the memory-efficient kernel. The recurrent baselines appear twice. The upper block runs them on optimized kernels: Mamba-2 on the SSD kernel of `mamba_ssm`, and DeltaNet and Gated DeltaNet on the chunked kernels of `flash-linear-attention`, which agree with the reference forms to within 0.4% relative error. The DeltaNet kernel does not accept fp32 inputs, so we ran it on bf16. Log-Linear uses the upstream kernels. The lower block runs the reference PyTorch implementations, as trained for Table 9. SMat decodes with the cached decoder of Section 3.3; softmax decodes from a preallocated KV cache, and the recurrences decode one step at a time. Before timing, every decoder is checked against the full forward pass. The largest relative error in fp32 is 5.3×10^{-4}, and the SMat decoder reproduces the prefill rows to 10^{-16} in fp64.

Table 10 sheds light on SMat's improvements over other attention variants. Up to $d = 3$, the hard route SMat is linear in time from end to end. From 16K to 64K its training step grows by 3.7–$4.3\times$,while softmax attention's grows by $15.4\times$. Prefill grows by 3.5–$4.0\times$ from 64K to 256K, against $16\times$ for softmax. Per-token decoding is flat at 0.25 ms from 1K to 256K, and 0.22 ms at batch 1. Against softmax attention this means 29–$65\times$ faster training steps at 64K, 210–$300\times$ faster prefill at 256K, $51\times$ faster decoding at 256K, and a cache $5.5\times$ ($d = 3$) to $43\times$ ($d = 2$) smaller than the KV cache. Against optimized linear-time kernels, SMat's Triton prefill is comparable: at 256K it matches DeltaNet and is faster than Gated DeltaNet. SMat training runs on unfused PyTorch operations and is 1.2–$1.8\times$ slower than those kernels at 64K for $d \leq 2$, and 2.6–$3.7\times$ slower for $d = 3$. It remains faster than the upstream Log-Linear kernels. SMat's decoding cache holds $B + 1$ summaries, so it exceeds the constant-size recurrent state. At short lengths it can also exceed the KV cache: at 1K, $d = 3$ uses 36 MB against 17 MB for softmax, and the order reverses by 4K. At $d = 4$ the $O(T^{2-3/d})$ long-range term dominates at these lengths. SMat is then slower than the linear-time kernels, but still $1.8\times$ faster than softmax in training at 64K and $18\times$ faster in prefill at 256K. Gating adds only linear work (Lemma 4). Our gated causal scan, however, is an unfused PyTorch loop over chunks, so gated training is about $20\times$ slower than ungated at 64K, while gated prefill and decoding stay within 2.5–$5\times$ and $1.4\times$ of ungated.

Model	Train (ms/step) 16K	64K	Prefill (ms) 64K	256K	Decode (ms/token)	Cache (MB)
SMat ($d = 1$)	12.9	48.1	11.5	45.6	0.25	0.5
SMat ($d = 2$)	14.1	51.6	11.3	44.5	0.25	99
SMat ($d = 3$)	25.1	107.0	18.3	63.8	0.25	774
SMat ($d = 4$)	192.8	1717.9	168.6	725.4	0.26	3439
Gated SMat ($d = 2$)	92.7	988.9	43.1	211.3	0.34	99
Gated SMat ($d = 3$)	104.0	1044.3	70.9	316.9	0.34	774
Softmax (SDPA)	204.4	3147.9	836.5	13410.2	12.90	4295
Mamba-2 (SSD kernel)	8.6	29.1	7.2	–[a]	0.22	0.3
DeltaNet (FLA, bf16)	8.9	31.7	11.7	46.5	0.24	0.3
Gated DeltaNet (FLA)	11.3	41.9	13.8	54.7	0.31	0.3
Log-Linear (upstream)	21.9	97.8	22.8	OOM[b]	–	–
Reference implementations, as trained for Table 9						
Mamba-2	66.0	683.0	36.3	175.2	0.22	0.3
DeltaNet	250.7	3230.4	85.4	396.4	0.24	0.3
Gated DeltaNet	279.0	3377.7	121.7	534.4	0.31	0.3
Log-Linear	OOM[c]	–	340.7[d]	–	–	–

Table 10: End-to-end cost of the subset-routing models: batch 64, fp32. SMat prefill uses the Triton kernels. Decode latency and cache size are at context length 256K. Optimized and reference recurrences share the same one-step decoder. [a] The SSD kernel exceeds a CUDA launch limit at batch 64 and length 256K. [b]The upstream port builds a dense $T \times T$ level table (512 GiB at 256K). [c]The reference Log-Linear builds the same dense table and runs out of memory in tr aining at 16K. [d]Value at 16K.

Multi-key subset recall. Keys and payloads are placed at random positions in the distant block, so the task cannot be solved positionally. The hyperplane read is used throughout, and the reported metric is exact-support match: a query counts as correct only when the retrieved support equals the requested key set. *LSH bucketing* (Kitaev et al., 2020) hashes keys into the same cells, but a query reads only its own cell rather than a hyperplane of cells. *Memory-matched linear attention* tests whether SMat's advantage is simply the larger state it keeps, since associative recall in efficient models is known to be limited by state size (Arora et al., 2023).

Model	$k = 1$ s0	s1	s2	$k = 2$ s0	s1	s2	$k = 3$ s0	s1	s2
Softmax	100.0	7.9	25.0	99.9	0.0	100.0	99.9	0.0	99.5
Mamba-2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
DeltaNet	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Gated DeltaNet	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Log-Linear	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Linear attention	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Linear attention (memory matched)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
LSH bucketing (121 buckets)	100.0	100.0	100.0	0.7	0.4	0.3	0.0	0.0	0.0
LSH bucketing (125 buckets)	100.0	100.0	100.0	1.8	1.2	2.1	0.0	0.0	0.0
SMat ($d = 2$)	100.0	100.0	100.0	11.7	14.0	16.0	0.9	1.6	1.4
SMat ($d = 3$)	100.0	100.0	100.0	99.9	99.9	99.7	35.5	33.1	34.0
SMat ($d = 4$)	100.0	100.0	100.0	100.0	99.9	99.9	99.0	99.0	99.0

Table 11: Per-seed exact-support accuracy (%) behind Table 1, after 12K steps on 2,048 fresh queries per run. SMat rows use rule-chosen hyperplane directions (see text).

To further examine these results, a network that ignores the query does best by predicting the base rate $k/8$ for every pair. That constant predictor has loss $H(k/8) = 0.377$, 0.562 and 0.662 nats for $k = 1, 2, 3$, and its logit $\log \frac{k}{8-k} < 0.5$, so it marks every pair absent. More generally, a constant output is all-absent or all-present under any threshold and therefore never equals a k-hot target, so it scores exactly 0 whether the threshold is placed on logits or on probabilities. All 54 linear-baseline runs end within 0.0012 nats of $H(k/8)$ (Figure 7). Their zeros therefore come from

models that never began to use the query. For reference, guessing k pairs uniformly at random scores $1/\binom{8}{k} = 12.5\%$, 3.6% and 1.8%.

Table 11 lists every run behind Table 1, and Figure 7 shows the training loss and exact-support accuracy of each seed. Softmax outcomes are bimodal. Each run starts on the constant-prior plateau with some runs leaving after 6K and 10K steps and reaching $\geq 99\%$ by the next evaluation. Seed 1 never leaves the plateau at $k = 2, 3$; at $k = 1$, seeds 1 and 2 had begun to leave it when training stopped. The large standard deviations of the softmax row therefore record whether a seed escaped the plateau within the budget. The other warmup and learning-rate settings we tried for softmax did no better (Table 12).

It is also important to note the difference between the rule-chosen and learned hyperplane schemas for the SMat matrix in this experiment. In the SMat rows of Table 1, the hyperplane direction at each answer row is set, during training and evaluation, by a fixed rule that picks a direction on which the hashed cells of all k requested keys agree whenever one exists. The baselines have no equivalent mechanism. This isolates what the mask can express. When a linear head predicts the direction from the query instead (seed 0 only), SMat scores 99.9, 60.1 and 44.6% at $d = 3$ and 100.0, 74.2 and 59.2% at $d = 4$ for $k = 1, 2, 3$. At $d = 2$ the hyperplanes are single points, so there is only one hyperplane direction and the two schemas coincide.

Softmax schedule, LR	$k = 1$	$k = 2$	$k = 3$
no warmup, 3×10^{-4} (reported)	100.0 / 7.9 / 25.0	99.9 / 0.0 / 100.0	99.9 / 0.0 / 99.5
1K-step warmup, 10^{-4}	100.0 / 0.0 / 13.4	0.0 / 0.0 / 0.0	0.0 / 0.0 / 0.0
1K-step warmup, 3×10^{-4}	100.0 / 1.5 / 13.4	0.0 / 0.0 / 2.8	99.8 / 0.0 / 0.0
1K-step warmup, 10^{-3}	0.0 / 0.0 / 100.0	0.0 / 0.0 / 0.0	0.0 / 0.0 / 0.0

Table 12: Softmax on multi-key subset recall at $T = 1024$: exact-support accuracy (%) for seeds 0 / 1 / 2 under each schedule tried. A run that leaves the constant-prior plateau reaches $\geq 99\%$; the runs between 1% and 25% had begun leaving it when training stopped. The reported row is the best schedule.

Model	$T = 128$	$T = 256$	$T = 1024$
Softmax	97.5	99.5	100.0
Mamba-2	99.2	0.0	0.0
DeltaNet	100.0	100.0	0.0
Gated DeltaNet	100.0	100.0	0.0
Log-Linear	99.9	93.6	0.0
Linear attention	94.8	35.6	0.0

Table 13: Positive control: exact-support accuracy (%) for $k = 1$, seed 0, with the harness, metric, model and 12K-step budget of Table 1 and only the context length changed (8 pairs throughout).

Table 14: Mamba-2 SMat results across three seeds. Mean (std).

Model	$k = 1$	$k = 2$	$k = 3$
Mamba-2	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
SMat ($d = 2$)	99.48 (0.72)	2.51 (3.54)	0.00 (0.00)
SMat ($d = 3$)	96.43 (4.65)	84.29 (19.86)	0.00 (0.00)
SMat ($d = 4$)	100.00 (0.00)	99.98 (0.02)	99.90 (0.14)

Table 14 shows the results of a learned Mamba-2–SMat on multi-key subset recall. The baseline Mamba-2 achieves zero exact-support accuracy under this training setup, whereas SMat with $d = 2$ succeeds primarily at $k = 1$, $d = 3$ substantially improves performance at $k = 2$, and $d = 4$ achieves near-perfect accuracy across all three settings. This qualitatively matches the original MKAR pattern, with increasing geometric dimension enabling accurate retrieval of larger requested subsets.

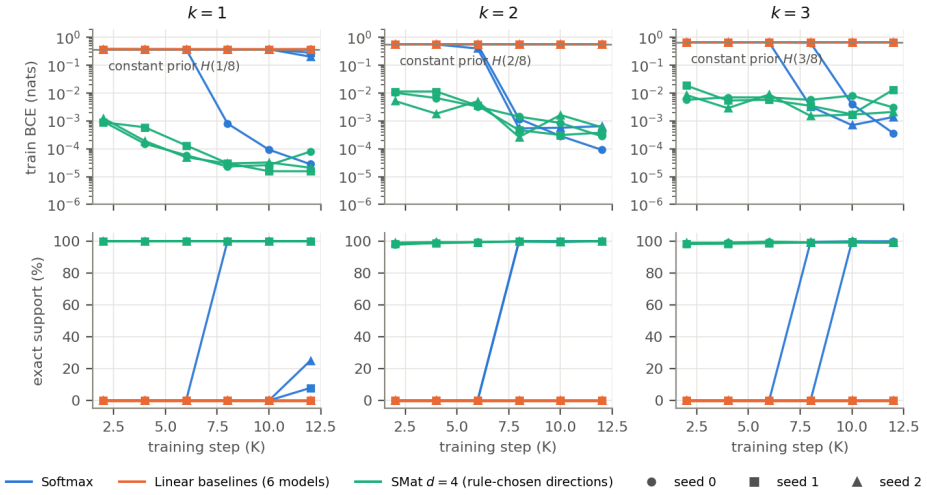

Figure 7: Multi-key subset recall at $T = 1024$, every seed. Top: training binary cross-entropy of the minibatch at each evaluation step (log scale); the dashed line is the constant-prior loss $H(k/8)$, and all 18 linear-baseline runs per panel lie on it. Bottom: exact-support accuracy on 2,048 fresh queries, evaluated every 2K steps.

MQAR. Run through the Zoology harness (Arora et al., 2023) with its data pipeline unchanged: training mixtures of 4, 8, 16, 32, and 64 key–value pairs at lengths 64–256 (100K examples for the 4-pair mixture, 20K for each of the others), and 1000 held-out examples per mixture. Evaluation batch size is 32. The GDN backbone uses one head at width 16 and two heads at widths 32 and 64, with value expansion 1, following Guo et al. (2025); 32 epochs is 707 optimizer steps per epoch. Log-Linear Attention uses the authors' released implementation with the same widths, head configuration, and value expansion as the corresponding backbone, and the same 32-epoch budget.

Joint context-key recall (JCKR). Models retrieve values from shuffled context-key-value records, with keys shared across contexts and values sampled uniformly from 16 symbols. Every context-key pair is queried in random order, with answers masked and cross-entropy applied only at query positions. We use $(C, K) \in \{(1, 4), (2, 8), (8, 16), (16, 16), (32, 16)\}$, yielding sequence lengths 64–3076, with 36000/2000/4000 train/validation/test examples per configuration. Two-layer models of width 64 train for 32 epochs using AdamW, learning rate 3×10^{-3}, cosine decay, and batch size 256 (Table 8). We report final-epoch validation accuracy averaged equally across configurations, with mean and standard deviation over three training seeds.

Model	Accuracy (%)
Plain GDN	53.80 (2.86)
GDN + SMat ($d = 3$)	**58.92** (**4.22**)
MoM-derived, profile-count matched	43.73 (1.48)
MoM-derived, matrix-storage matched	35.10 (3.82)

Table 15: Shared-key joint recall: final-test accuracy averaged equally over the five evaluated binding loads. Mean and std reported across three seeds.

Furthermore, we compare SMAT with two variants derived from Mixture-of-Memories (MoM) Du et al. (2026), both using Gated DeltaNet updates, top-4 routing, and no shared memory. At each sequence length, the profile-count variant has one independent memory matrix per SMAT profile, while the matrix-storage variant matches the combined FP32 storage of SMAT's profile and derived summary matrices. All memory matrices are 16×16. We retain the same data splits, model width, layer and head counts, and 32-epoch training recipe.

Model	16K	32K
Mamba-2	3.752	3.803
SMat $d = 2$	3.743	3.788
SMat $d = 3$	3.744	3.787
SMat $d = 4$	3.741	3.788
Softmax	3.797	3.910

(a) 300M training tokens

Variant	GDN	Mamba-2
Baseline	23.165	24.114
SMat ($d = 2$)	23.097	24.117
SMat ($d = 3$)	23.214	24.182

(b) 750M training tokens

Table 16: PG-19 language modeling results. Left: negative log-likelihood (NLL) per token after 300M training tokens with eight layers and width 384. Right: validation perplexity after 750M training tokens at 16K context (seed 123). Lower is better.

PG-19. Books are tokenized with the GPT-2 BPE vocabulary and packed into fixed-length windows; all arms see the same data order. The transformer arm is attention plus a $4\times$ MLP with eight heads, sized so that its non-embedding parameter count is comparable to SMat's. An auxiliary load-balancing term with coefficient 0.01 is applied to the routing hash and annealed over the first 1000 steps. Evaluation is per-token negative log-likelihood on the PG-19 test split (100 books, 5×10^6 tokens) at the training context length. Peak memory is roughly 15 GB per arm at 16K and 30 GB at 32K. [†]

[†]Due to academic compute constraints, each model configuration in Table 16 was trained with a single seed.