FREESOLO: LINEAR-TIME SPARSE ATTENTION WITH ZERO-SHOT LENGTH EXTRAPOLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Linear-time sequence models typically store long-context information in a recurrent state. A fixed-size state limits how much information can be stored, while hybrid architectures that add full attention for better recall incur quadratic sequence cost. Alternatively, sparse attention with continued pretraining (CPT) can extend the effective context length. Generalization beyond the trained length remains challenging, however, and CPT can be costly. We introduce FREESOLO, a native sparse attention mechanism with a short-range sliding window branch and a long-range branch that selects keypoint cells for direct token attention. Its linear-time variant, FS-L, learns to prune the prefix before query-dependent selection through short CPT. FreeSolo demonstrates better zero-shot length extrapolation than the baselines without long-context fine-tuning.

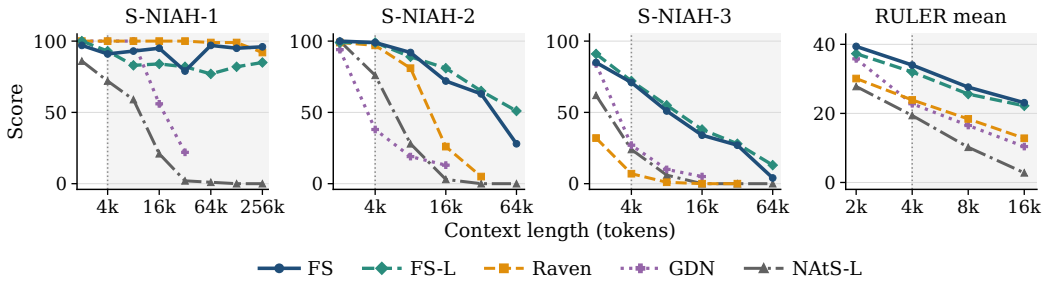

Figure 1: **Zero-shot RULER extrapolation at** 800**M.** Models train at 4k (dotted line); shading marks extrapolation. 100 examples per task and length.

1 INTRODUCTION

Long-context language modeling requires efficient attention and the ability to use information beyond the training context. Full attention has quadratic cost in sequence length (Vaswani et al., 2017). Linear-attention and state-space models reduce this cost by compressing history into a recurrent state of fixed size (Katharopoulos et al., 2020; Gu & Dao, 2023; Dao & Gu, 2024). Sparse attention preserves access to individual tokens, but methods such as Native Sparse Attention (NSA) (Yuan et al., 2025) and Mixture of Block Attention (MoBA) (Lu et al., 2025) still compare each query with a growing set of blocks. Neither recurrent compression nor sparsity alone ensures that a model trained on short sequences can retrieve information from longer ones. Continued pretraining (CPT) on longer sequences can extend this range (Fu et al., 2024), at an additional training cost. Recent models such as Raven (Afzal et al., 2026) and NAtS-L (Deng et al., 2026) target efficient sequence modeling with length extrapolation.

In this work, we propose FREESOLO (FS), a native sparse attention mechanism with zero-shot length extrapolation. The main idea is to combine range separation with keypoint detection. A local branch applies RoPE (Su et al., 2021) within a fixed sliding window. A long-range branch uses cell summaries to identify keypoints. Each query selects top-K cells and attends directly to their tokens without positional rotations (Figure 2).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

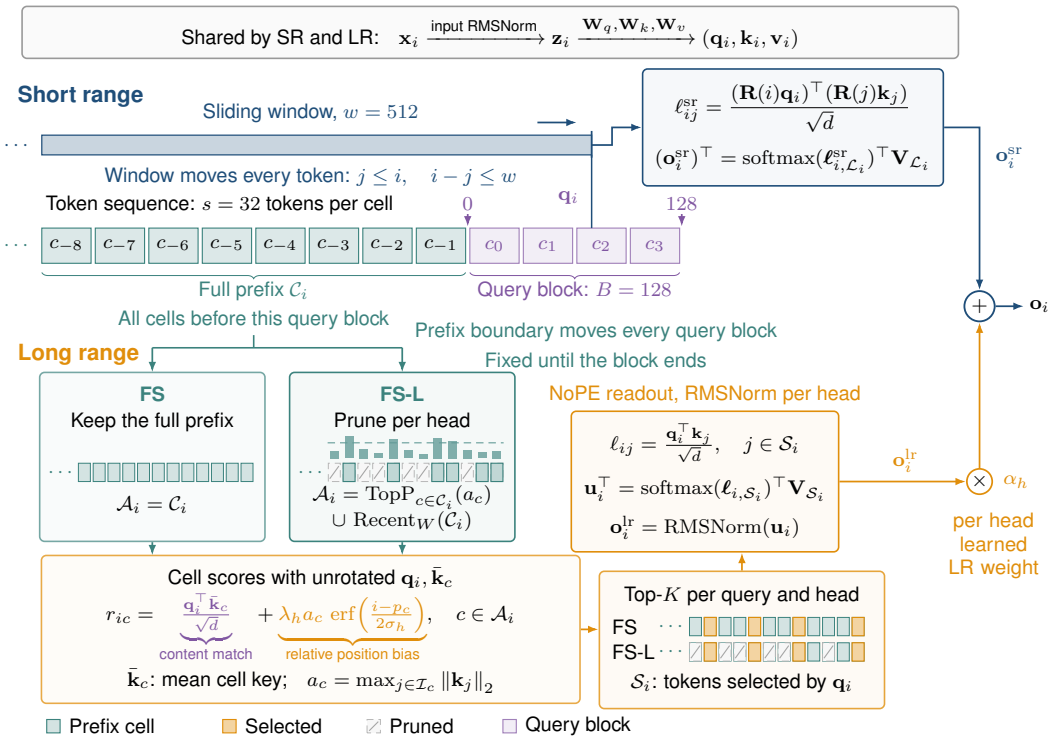

Figure 2: **FreeSolo attention.** FS scores the full prefix; FS-L prunes per head, independently of the query. Both select $K = 4$ cells per query and head. We use $P = 64$ high-norm cells, $W = 8$ recent cells, cell size $s = 32$, query blocks of $B = 128$ tokens, and SR radius $w = 512$.

FS scores the full prefix. FreeSolo-Linear (FS-L) first retains high-norm cells and a recent window independently of the query. Incremental updates bound both routing and readout, giving $O(N)$ attention time and bounded attention state. We obtain FS-L through short continued pretraining of FS at the same 4096-token context. FreeSolo generalizes on the harder S-NIAH-2/3 tasks, with stronger recall than standalone Raven at extended lengths (Figure 1). Both FS and FS-L also retrieve single needles at 256k, $64\times$ the training length. We provide an information-bottleneck view (Tishby et al., 2000) that separates locating evidence from retaining it before a future query.

2 Preliminaries

Attention. For one head, let $\mathbf{q}_i, \mathbf{k}_i, \mathbf{v}_i \in \mathbb{R}^d$ be the query, key, and value at position $i \in \{0, \dots, N-1\}$. Define $\ell_{ij} = \mathbf{q}_i^\top \mathbf{k}_j / \sqrt{d}$ and $\Delta = i - j$. For a nonempty token set $\mathcal{J}$, collect the logits in $\boldsymbol{\ell}_{i,\mathcal{J}}$ and matching value rows $\mathbf{v}_j^\top$ in $\mathbf{V}_{\mathcal{J}}$. The readout is

$$\mathbf{o}_i^\top = \mathrm{softmax}(\boldsymbol{\ell}_{i,\mathcal{J}})^\top \mathbf{V}_{\mathcal{J}} = \frac{\sum_{j\in\mathcal{J}} \exp(\ell_{ij}) \mathbf{v}_j^\top}{\sum_{j\in\mathcal{J}} \exp(\ell_{ij})}. \tag{1}$$

Full causal attention uses $\mathcal{J} = \{0, \dots, i\}$ and costs $O(N^2 d)$ per sequence or $O(Nd)$ per cached decoding step. FlashAttention computes the same readout with $O(Nd)$ memory (Dao et al., 2022).

Sparse attention. Sparse attention restricts each query to $\mathcal{J}_i \subseteq \{0, \dots, i\}$, fixed or query dependent. If $|\mathcal{J}_i| \le m$, the readout costs $O(Nmd)$ per sequence, plus selection.

Length extrapolation. Zero-shot length extrapolation evaluates at $N > L$ after training at context length L, with fixed weights and no training on longer contexts. We measure retrieval separately from perplexity, whose token average can obscure errors requiring distant context (Fang et al., 2024).

3 METHOD

3.1 RANGE SEPARATION

FreeSolo combines local window attention with sparse long-range attention in each head (Figure 2). The local branch applies RoPE within a fixed window. The long-range branch selects top-K cells and applies NoPE attention (Kazemnejad et al., 2023) to their tokens, using unrotated queries and keys. This branch is sparse in both FS and FS-L; FS-L also prunes the prefix before query-dependent scoring. Both branches share projections of the layer's RMS-normalized hidden state $\mathbf{z}_i$ (Zhang & Sennrich, 2019): $\mathbf{q}_i = \mathbf{W}_q\mathbf{z}_i$, $\mathbf{k}_i = \mathbf{W}_k\mathbf{z}_i$, and $\mathbf{v}_i = \mathbf{W}_v\mathbf{z}_i$. We combine the outputs as

$$\mathbf{o}_i = \mathbf{o}_i^{\mathrm{sr}} + \alpha_h \mathbf{o}_i^{\mathrm{lr}}, \tag{2}$$

where α_h is learned per head. Each branch normalizes its own attention, and their supports may overlap. We concatenate the head outputs, project by $\mathbf{W}_o$, and add the layer input. A second RMSNorm and SwiGLU feed-forward residual (Shazeer, 2020) complete the layer.

3.2 LOCAL BRANCH

Let $\mathcal{L}_i = \{j : 0 \le j \le i,\ i - j \le w\}$, where w is a look-back radius, so a full window contains $w + 1$ tokens. Let $\mathbf{R}(i)$ be the RoPE rotation at position i (Su et al., 2021). We use the local score

$$\ell_{ij}^{\mathrm{sr}} = \frac{(\mathbf{R}(i)\mathbf{q}_i)^\top(\mathbf{R}(j)\mathbf{k}_j)}{\sqrt{d}} = \frac{\mathbf{q}_i^\top \mathbf{R}(j-i)\mathbf{k}_j}{\sqrt{d}}, \tag{3}$$

and collect these logits over $\mathcal{L}_i$ to compute

$$(\mathbf{o}_i^{\mathrm{sr}})^\top = \mathrm{softmax}(\boldsymbol{\ell}_{i,\mathcal{L}_i}^{\mathrm{sr}})^\top \mathbf{V}_{\mathcal{L}_i}. \tag{4}$$

We use $w = 512$ at both training and evaluation. Increasing the context therefore introduces no larger relative offsets into this branch.

3.3 LONG-RANGE BRANCH

The long-range branch provides a sparse, content-addressable memory beyond the local window. Storing contextualized token key/value pairs allows later queries to retrieve individual details directly (Wu et al., 2022; Xiao et al., 2024a). Mean keys guide cell selection; the readout uses the selected tokens' keys and values. FS scores the full prefix before the query block. FS-L first prunes this prefix by key norm and recency, separating which cells remain available from which cells a query reads. This separates memory capacity from per-query read bandwidth (Appendix D.2).

Cells. A cell is a group of s consecutive token key/value pairs, selected as a unit by the router. Using zero-based indices, cell c contains positions $\mathcal{I}_c = \{cs, \ldots, (c+1)s - 1\}$. Its summaries are

$$\bar{\mathbf{k}}_c = \frac{1}{s}\sum_{j \in \mathcal{I}_c} \mathbf{k}_j, \qquad p_c = (c + \tfrac{1}{2})s, \qquad a_c = \max_{j \in \mathcal{I}_c} \|\mathbf{k}_j\|_2. \tag{5}$$

The mean key summarizes content for routing, p_c lies halfway between the cell boundaries, and the largest key norm supplies a query-independent priority. The raw keys and values remain available for token-level attention after selection.

Query blocks. A query block groups B/s consecutive cells, covering B token positions, with B a multiple of s. The full prefix is shared across heads and queries in a block. FS-L reuses each head's retained pool throughout the block, amortizing pruning and gathering. Selection and readout are per query and head. We use $B = 128$ and $s = 32$, so each block contains four cells. The block containing query i starts at $b_i = B\lfloor i/B \rfloor$.

Causal availability. The full prefix $\mathcal{C}_i$ contains all cells before the query-block start:

$$\mathcal{C}_i = \{c \in \mathbb{Z}_{\ge 0} : (c+1)s \le b_i\}. \tag{6}$$

Every cell in this prefix contains only past tokens for every query in the block. The prefix stays fixed until all B queries have finished (Figure 2). At the next boundary, it includes the completed block, and FS-L updates its retained cells. The local window covers the intervening past because $w = 512 \ge B - 1$; the first block has no long-range cells.

Cell scoring. We combine mean-key content routing, also used in MoBA (Lu et al., 2025), with a norm-weighted relative position bias:

$$r_{ic} = \frac{\mathbf{q}_i^\top \bar{\mathbf{k}}_c}{\sqrt{d}} + \lambda_h \, a_c \, g_h(i - p_c), \qquad c \in \mathcal{C}_i, \tag{7}$$

where $g_h(\Delta) = \mathrm{erf}(\Delta/(2\sigma_h))$, $\sigma_h > 0$, and $\lambda_h \geq 0$. The content term uses unrotated queries and keys. The erf prior encodes relative distance in a translation-invariant way and supplies a norm-scaled bias to cell scoring. All eligible offsets satisfy $i - p_c > 0$. Ranking these scores and selecting only the top-K cells induces sparsity: only tokens in the selected cells enter the long-range readout. FS scores the full prefix; FS-L first prunes it by key norm.

Query-independent pruning. Each head prunes independently using its own projected key norms. FS-L retains the W most recent cells and the P cells with largest a_c:

$$\mathcal{A}_i = \mathrm{Recent}_W(\mathcal{C}_i) \;\cup\; \mathrm{TopP}_{c \in \mathcal{C}_i}(a_c), \qquad |\mathcal{A}_i| \leq W + P. \tag{8}$$

The union removes duplicates. The priority favors cells containing a large-norm key, while the recent window preserves nearby context. Projected keys have varying norms; we do not normalize them individually. The RMSNorm after the long-range readout normalizes the output, not these priorities.

Key norm supplies a query-independent priority, not a guarantee of relevance to every query. CPT adapts the model to this restricted access pattern. Appendix D.3 measures how much pruning compresses the prefix and which cells remain available for retrieval. We use $P = 64$ and $W = 8$.

Per-query selection. FS uses $\mathcal{A}_i = \mathcal{C}_i$; FS-L uses Eq. (8). Each head evaluates Eq. (7) on its own $\mathcal{A}_i$ and selects up to $K = 4$ cells per query. Different heads can select different cells, and selections can change at every token within a query block. Let $\mathcal{T}_i(\mathcal{A}_i)$ contain the $\min(K, |\mathcal{A}_i|)$ highest-scoring cells, and collect their token indices as $\mathcal{S}_i = \bigcup_{c \in \mathcal{T}_i(\mathcal{A}_i)} \mathcal{I}_c$.

Token readout. The long-range output is

$$\mathbf{u}_i^\top = \mathrm{softmax}(\boldsymbol{\ell}_{i,\mathcal{S}_i})^\top \mathbf{V}_{\mathcal{S}_i}, \qquad \mathbf{o}_i^{\mathrm{lr}} = \mathrm{RMSNorm}(\mathbf{u}_i). \tag{9}$$

This is NoPE attention: the erf prior affects cell scoring only, while the readout uses unmodified token keys and values, with no positional bias in its logits. The softmax is normalized jointly over all selected tokens. RMSNorm acts over the d head features (Zhang & Sennrich, 2019). Both outputs in Eq. (9) are zero when $\mathcal{S}_i$ is empty.

Training. We obtain FS-L by continued pretraining of FS with pruning enabled. Both stages use the standard next-token cross-entropy objective on 4096-token sequences. CPT resets the optimizer and learning-rate schedule (Appendix A) and requires no longer training contexts. Causal masks allow all query blocks to be evaluated in parallel. Hard top-P and top-K indices receive no gradient; shared projections and contextual states learn through local attention and selected-token readout. The routing-only scalars receive no gradient in this configuration and remain at $\sigma_h = 4$ and $\lambda_h = 1$.

Training with pruning from scratch is also possible, but CPT gives better overall retrieval, particularly on the harder tasks. The 340M from-scratch model retains strong S-NIAH-1 recall, whereas the tested 800M run retrieves poorly. Starting from FS provides useful representations before imposing the pruning constraint. This motivates the two-stage recipe, although the runs do not isolate the cause of the from-scratch difference. Appendix D.4 gives the comparisons and variation across CPT runs.

Inference. Prompt prefill uses the same causal operator and populates each layer's raw key/value cache and cell summaries. During decoding, each new token appends its key and value; a completed cell supplies its mean key and maximum key norm. The local window advances every token. The long-range prefix advances only at query-block boundaries, when FS-L also updates its retained set. Within a block, each query scores the available cells, selects its own top-K, and applies Eq. (9) before the branch combination in Eq. (2). FS-L's streaming cache merges newly eligible cells into the retained set and discards unused KV once it leaves the recent, local, and unfinished-block buffers.

3.4 COMPUTATIONAL COST

FS requires $O(N^2/s)$ attention work per sequence. FS-L achieves $O(N)$ by pruning cells independently of the query and maintaining a bounded pool incrementally. These costs assume fixed head dimensions and routing budgets.

Cell statistics cost $O(Nd)$ per head. Each query scores $M \simeq N/s$ cells, giving FS a cost of $O(N^2d/s+N(w+Ks)d)$ including local and selected-token readout. Query blocks share availability, not scores; B does not reduce the quadratic term. FS-L incrementally merges newly eligible cells into its top-P set. Fixed key-norm scores prevent discarded cells from re-entering. A heap costs $O(M \log(P+1))$ in total; bounded-array merges are linear at fixed budgets.

Let $C = P + W$. Scoring the retained cells costs $O(NCd)$, and attention over gathered selected tokens costs $O(NKsd)$. With incremental retention and the local branch, the sequence cost is

$$O(Nd + M \log(P+1) + N(C + w + Ks)d) \,. \tag{10}$$

At fixed budgets, decode work is bounded, including amortized retention updates. The streaming implementation keeps $O(((P+W)s + w + B + s)d)$ token state per head plus indices and counters, computes rotations on demand, and prefills in fixed chunks (Appendix D.6).

The full-history reference used for accuracy evaluations and Table 5 stores $O(Nd)$ KV and rescans priorities at block boundaries, adding $\Theta(N^2/(Bs))$ work. Its masked readout costs $O(N(P+W)sd)$; the indexed path reads selected tokens. Streaming-cache measurements are in Appendix C.

3.5 POSITION DEPENDENCE

Long-range token attention is NoPE: the erf prior encodes relative position only in cell scoring, leaving token keys, values, and attention logits unmodified. Both $g_h(i - p_c)$ and local RoPE scores depend on relative offsets, so shifting tokens, cell partitions, and query blocks together preserves the operator at fixed content vectors (Appendix D.1). Local offsets stay within w; the erf factor approaches a constant. These properties avoid new long-range rotary offsets but do not guarantee generalization to added content. Section 5 discusses selection and pruning; we do not attribute extrapolation to erf's nearly saturated distance dependence.

4 EXPERIMENTS

4.1 SETUP

We train LLaMA-style decoders (Touvron et al., 2023) on FineWeb-Edu (Penedo et al., 2024), using a 32k LLaMA tokenizer and a 4096-token context. The two FreeSolo configurations have 341M and 743M parameters, with pretraining budgets of 15B and 32B tokens. The smaller run uses about 2.2 passes over a 6.7B-token subset; the larger run draws from a larger corpus. Training uses bf16 and AdamW (Loshchilov & Hutter, 2019); see Appendix A for architecture and optimizer settings.

FS-L uses 0.73B additional CPT tokens at 340M and 1.57B at 800M, about 5% of the respective pretraining budgets, all at 4096-token context. Appendix D.4 compares continuation with and without pruning at both scales, two adaptation seeds at 340M, and training from scratch.

Baselines. TF++ is the full-attention Transformer++ reference (Yang et al., 2025b) with RoPE; SWA uses a 512-token RoPE window. NSA (Yuan et al., 2025) and MoBA (Lu et al., 2025) select blocks for sparse token attention. GDN (Yang et al., 2025b) uses a recurrent gated delta rule, and NAtS-L (Deng et al., 2026) combines GDN with retained-token softmax attention. InfLLM (Xiao et al., 2024a) adds inference-time retrieval to the TF++ checkpoint; we evaluate it on S-NIAH-1/2/3.

Raven (Afzal et al., 2026) writes sparsely to key/value memory slots and reads them with softmax. We use the standalone 256-slot model as the linear-time Raven baseline. Raven-full has stronger published S-NIAH-2/3 recall, but its full NoPE attention costs $O(N^2)$, so we exclude it from this linear-time comparison. Its SWA hybrid is linear at fixed window size but has weaker recall than Raven-full (Afzal et al., 2026, Table 9). Baselines share the corpus, tokenizer, and training context within each scale. The tables list parameter counts; Appendix A records training budgets and settings.

Complexity. Table 1 reports attention complexity at fixed head dimensions and budgets. For FS, MoBA, and NSA, the table uses $O(N^2/B)$, with B a fixed architecture-specific reduction factor. Reference kernels may perform additional dense work; our InfLLM lookup is exhaustive. FS-L is linear with incremental retention (Section 3.4), as are Raven, GDN, and SWA at fixed state or window size. NAtS-L costs $O(NR)$, where R is the data-dependent number of tokens retained for softmax. A fixed retention fraction gives quadratic cost (Deng et al., 2026).

Evaluation. We evaluate NVIDIA RULER (Hsieh et al., 2024) with 100 examples per task and length, greedy decoding, and official 0–100 scores. Suite means cover 12 tasks at 2k/4k and 13 at 8k/16k, which add QA-2. We omit further long-context runs when a model's retrieval has already failed at shorter extrapolation lengths. Appendix B reports all task scores; Appendix A gives the protocol.

4.2 Zero-shot extrapolation for long-context retrieval

Raven, NAtS-L, and GDN retain retrieval ability beyond the training context, whereas MoBA and NSA do not sustain recall in our evaluations (Table 1). FreeSolo's advantage is clearest on the harder S-NIAH-2/3 tasks (Figure 1). At 800M and 16k, four times the training length, FS-L scores $81/38$, compared with $26/0$ for Raven and $72/34$ for FS. FS-L reaches $51/13$ at 64k, compared with $28/4$ for FS and $0/0$ for NAtS-L. Relative to FS, the 800M FS-L checkpoint trades some S-NIAH-1 recall for higher S-NIAH-2/3 scores at 16k–64k, although absolute accuracy still declines with length.

FS and FS-L also outperform Raven and NAtS-L on the full RULER mean at both scales. At 800M/16k, the means are $23.1/22.2$ for FS/FS-L, versus $12.8/2.8$ for Raven/NAtS-L. The gains extend beyond single needles: FS-L scores 24 on MK-NIAH-1 versus 3 for both linear baselines, and 16 on MV-NIAH versus $9/0.5$ for Raven/NAtS-L (Table 3). Some tasks, including variable tracking, remain difficult for all models even within the training context.

Matched continuations distinguish adaptation to pruning from additional training (Appendix D.4). On the same 50 prompts at 800M/16k, FS continuation scores $84/38$ on S-NIAH-2/3 and FS-L scores $80/42$. Thus short CPT preserves much of the harder-task retrieval while reducing attention cost from quadratic to linear. These controls use each model's native inference budget; the gain is efficient retrieval, without requiring pruning itself to improve accuracy.

S-NIAH-1 tests retrieval over a much longer range. At 340M, FS maintains 98–100 through 256k and FS-L scores 84 at 256k; Raven falls to 20 at 131k. At 800M, FS and FS-L score $96/85$ at 256k, compared with zero for NAtS-L. Raven also extrapolates well on this task, scoring 99–100 through 131k and 92 at 256k. S-NIAH-1 alone misses FreeSolo's advantage on harder tasks.

4.3 Short-context quality

Table 2 reports eight zero-shot tasks with task definitions from the Language Model Evaluation Harness (Gao et al., 2024). We use `acc_norm` for HellaSwag, ARC-c, and OBQA, and `acc` for the other tasks; Avg is their arithmetic mean. At 340M, FS averages 51.9 and FS-L 51.0, compared with 51.8 for TF++, 51.4 for Raven, and 50.8 for NAtS-L. At 800M, FS reaches 56.5, close to TF++'s 56.8 and SWA's 56.2, while FS-L reaches 55.2 and Raven 54.6. The reported FS-L checkpoints retain similar short-context quality at both scales.

4.4 Empirical scaling

To verify linear scaling empirically, we increase context from 4k to 256k on one A100-40GB. FS-L's attention buffers remain constant at 766 MiB for the 340M model and 1149 MiB for the 800M model. CUDA-graph decoding runs at about 89 and 75 tokens/s, respectively, including retention updates. Prefill throughput stays near 2.1–2.2K tokens/s, so processing time grows approximately linearly with input length.

The accuracy tables use a full-history cache; Appendix C compares the two implementations on shared prompts.

The full-history reference prefills at 34.6–44.2K tokens/s but keeps a growing KV cache. Training at 340M/4k is also slower than full attention: 98.4K versus 303K tokens/s across eight GPUs.

Table 1: **RULER accuracy** (0–100). All models train at 4k; results at 8k and above test zero-shot extrapolation. Means cover available tasks; cost is defined in Section 4.1. Bold/underline mark the best/second-best scores within each block; dashes denote unavailable results.

			S-NIAH-1								S-NIAH-2						S-NIAH-3						Mean			
Model	Params	Complexity	2k	4k	8k	16k	32k	64k	131k	256k	2k	4k	8k	16k	32k	64k	2k	4k	8k	16k	32k	64k	2k	4k	8k	16k
TF++	335M	$O(N^2)$	**95**	57	0	0	–	–	–	–	**91**	**24**	0	0	–	–	38	**1**	0	0	–	–	**26.2**	**13.3**	**0.2**	**0.0**
InfLLM	335M	$O(N^2)$	94	**58**	**13**	**9**	–	–	–	–	89	23	**8**	**2**	–	–	**39**	**1**	**2**	**1**	–	–	–	–	–	–
FS	341M	$O(N^2/B)$	**99**	**99**	**100**	**100**	**98**	**100**	**98**	**100**	**100**	**77**	**41**	**17**	–	–	58	**14**	**7**	**0**	–	–	**30.8**	**23.2**	**16.5**	**11.8**
MoBA	341M	$O(N^2/B)$	76	12	0	0	–	–	–	–	83	23	0	0	–	–	**59**	2	0	**0**	–	–	20.6	7.1	0.0	0.0
NSA	513M	$O(N^2/B)$	43	23	0	0	–	–	–	–	48	18	0	0	–	–	1	0	0	**0**	–	–	13.0	7.1	–	–
FS-L	341M	$O(N)$	99	87	89	**87**	65	**85**	**85**	**84**	**99**	**72**	**39**	**18**	–	–	**70**	**16**	**8**	**1**	–	–	**32.3**	**21.4**	**15.5**	**10.7**
Raven	357M	$O(N)$	95	**96**	**93**	81	**78**	41	20	–	92	70	36	3	–	–	3	0	0	0	–	–	20.5	17.8	14.0	8.5
NAtS-L	380M	$O(NR)$	**100**	94	37	19	0	0	0	–	89	22	20	3	–	–	36	6	1	0	–	–	25.6	15.3	7.4	4.0
GDN	335M	$O(N)$	95	83	35	19	–	–	–	–	83	20	18	3	–	–	0	0	0	0	–	–	21.6	14.2	7.0	3.6
SWA	341M	$O(N)$	18	15	7	3	–	–	–	–	31	21	6	2	–	–	25	2	3	0	–	–	14.6	9.3	4.5	4.0
≈800*M scale*																										
TF++	743M	$O(N^2)$	**100**	95	0	0	–	–	–	–	**100**	**94**	0	0	–	–	67	**50**	0	**0**	–	–	**46.3**	**37.5**	**0.0**	**0.0**
InfLLM	743M	$O(N^2)$	**100**	**96**	**20**	**4**	–	–	–	–	**100**	89	**12**	**4**	–	–	**77**	**50**	**4**	**0**	–	–	–	–	–	–
FS	743M	$O(N^2/B)$	97	**91**	**93**	**95**	**79**	**97**	**95**	**96**	**100**	**99**	**92**	**72**	**63**	**28**	**85**	**71**	**51**	**34**	**27**	**4**	**39.4**	**34.0**	**27.6**	**23.1**
NSA	1128M	$O(N^2/B)$	**100**	75	5	–	–	–	–	–	98	46	3	–	–	–	30	7	0	–	–	–	35.6	23.5	2.3	–
FS-L	743M	$O(N)$	**100**	93	83	84	82	77	82	85	99	**99**	**89**	**81**	**65**	**51**	**91**	**72**	**55**	**38**	**28**	**13**	**37.3**	**32.0**	**25.6**	**22.2**
Raven	726M	$O(N)$	**100**	**100**	**100**	**100**	**100**	**99**	**99**	**92**	99	97	81	26	5	–	32	7	1	0	0	–	30.1	23.9	18.4	12.8
NAtS-L	793M	$O(NR)$	86	72	59	21	2	1	0	0	**100**	76	28	3	0	0	62	24	6	0	0	0	27.8	19.4	10.2	2.8
GDN	694M	$O(N)$	**100**	**100**	**100**	56	22	–	–	–	94	38	19	13	–	–	84	27	10	5	–	–	35.8	22.8	16.5	10.4
SWA	743M	$O(N)$	18	15	7	3	–	–	–	–	31	21	6	2	–	–	19	3	1	0	–	–	15.6	10.3	5.5	4.0

Table 2: **Zero-shot downstream accuracy** (0–100). Bold and underline mark the best and second-best scores within each scale.

Model	Params	HellaS.	PIQA	ARC-e	ARC-c	WinoG.	OBQA	SciQ	BoolQ	Avg
GDN	335M	39.2	66.8	55.8	**28.2**	51.9	33.0	**83.2**	59.8	**52.2**
FS	341M	38.9	66.8	**57.6**	25.9	50.9	32.4	81.2	**61.1**	51.9
TF++	335M	39.2	66.0	56.2	28.1	50.6	32.8	81.5	59.6	51.8
Raven	357M	37.6	65.2	55.8	27.6	51.2	32.2	80.6	60.8	51.4
SWA	341M	38.8	66.2	56.4	26.7	50.6	30.8	80.4	**61.1**	51.4
FS-L	341M	38.4	66.3	55.0	25.3	51.0	31.4	79.8	60.4	51.0
NAtS-L	380M	**39.9**	**67.6**	50.0	27.4	**52.3**	**34.0**	75.1	59.7	50.8
MoBA	341M	33.3	61.8	43.4	26.2	51.3	29.0	65.1	52.2	45.3
NSA	513M	32.3	62.6	43.8	24.9	51.9	28.8	66.4	45.0	44.5
≈800*M scale*										
TF++	743M	48.9	**71.0**	63.8	**34.1**	53.8	34.0	87.2	**61.9**	**56.8**
FS	743M	49.5	70.0	64.6	31.7	**56.0**	34.4	86.7	58.8	56.5
GDN	694M	47.4	69.1	**64.8**	32.8	53.8	37.0	88.1	58.2	56.4
SWA	743M	**49.6**	68.7	63.5	31.6	55.1	34.4	85.2	61.8	56.2
NSA	1128M	47.4	68.1	63.8	31.7	54.3	32.4	**88.8**	61.4	56.0
FS-L	743M	48.1	69.0	62.6	31.8	53.6	33.6	85.4	57.5	55.2
NAtS-L	793M	44.8	68.2	63.0	29.0	52.8	**37.4**	85.3	59.2	55.0
Raven	726M	45.8	68.7	63.4	31.7	54.0	35.8	84.4	53.2	54.6

The current kernels therefore demonstrate linear scaling. Faster kernels for incremental pruning, gathering, and sparse readout are future work; throughput also depends on memory traffic and hardware utilization (Appendix C). Reducing P to 16 cuts 340M attention storage to 320 MiB with similar decode speed, but accuracy varies non-monotonically with P (Appendix D.5).

5 Length extrapolation and learned retention

We analyze FreeSolo's length generalization through an information-bottleneck view. FS routes through cell summaries; FS-L further restricts memory before future queries arrive. Appendix D connects this view to the ablations and retrieval diagnostics.

Position dependence. Position handling affects how attention transfers to longer sequences (Press et al., 2022; Kazemnejad et al., 2023; Chi et al., 2023). FS uses RoPE only within trained local offsets and NoPE for long-range token attention, so longer contexts introduce no untrained rotary offsets. Added tokens can still change representations and introduce competing cells; the ablations below examine these design choices.

The information bottleneck in FS. The router identifies useful cells from their mean keys and maximum key norms. Token attention then reads the selected cells' original keys and values. We hypothesize that learning to route through these summaries encourages recognizable keypoints. Both query alignment and the norm bonus affect selection.

Each head reads at most Ks token values, regardless of prefix length. RMSNorm rescales the readout but cannot remove interference from irrelevant values; sparse selection controls which values can contribute. On a shared 256k prompt, both variants at both scales select the needle cell over $166{,}000$ tokens away. These traces show distant access, without establishing which reads determine the answer (Appendix D.2).

The additional bottleneck in FS-L. FS-L discards cells before future queries arrive. This can tolerate growing background when useful information remains concentrated in the retained state. Many independently queryable facts put more pressure on that state: usefulness depends on which questions will be asked later (Appendix D.5).

CPT adapts the model to this restriction. At 340M/4k, it recovers S-NIAH-1 from 62 under immediate pruning to 78–86; unpruned continuations retain 100 (Appendix D.4). FS-L retains 14.3% of eligible cells at 340M/16k. We separately measure whether the source cell is retained, whether the query selects it, and whether the model answers correctly (Appendix D.3). Keeping a cell does not by itself ensure that it is used successfully. Appendix D.4 explains how the model learns despite the hard indices.

Budgets and ablations. P sets the high-norm cells retained per head; K sets the cells read per query and head. Larger K cannot recover a pruned cell; larger P expands availability but need not improve answers. We test adapted budgets and retrieval among several independently queryable facts (Appendix D.5).

The local NoPE control changes both rotation and local bias. It preserves S-NIAH-1 extrapolation but reduces downstream accuracy from 51.9 to 38.3. Removing the long-range norm-weighted erf term retains strong S-NIAH-1 recall, worsens S-NIAH-2, and improves S-NIAH-3 (Appendix D.1). For eligible cells, $g_h \geq \mathrm{erf}(2) \simeq 0.9953$; replacing it by one preserves over 99.99% of selected cells on fixed traces. The term acts mainly as a norm bonus in this configuration. These results do not establish a generalization benefit from the erf distance dependence (Appendix D.1).

6 Related Work

Selective recurrent memory. Mamba, Mamba-2, and GDN compress history through input-dependent recurrent updates (Gu & Dao, 2023; Dao & Gu, 2024; Yang et al., 2025b). GSA and Raven read key/value slots with softmax (Zhang et al., 2024; Afzal et al., 2026); Raven also uses sparse writes. FS-L retains token cells for direct readout.

Mixture-of-Experts (MoE) routes tokens to parameter experts (Shazeer et al., 2017). Mixture-of-Memories (MoM) routes them to recurrent memory states (Du et al., 2025). FreeSolo routes each head to stored token cells. All heads remain active; sparsity is over the cells they access.

Native block-sparse attention. MoBA (Lu et al., 2025) scores blocks by query–mean-key dot products and applies softmax over the selected blocks, always including the causal current block. NSA (Yuan et al., 2025) derives block-selection scores from compressed attention and combines compressed, selected-token, and sliding-window outputs. Heads within each grouped-query attention group share selected blocks.

FS combines mean-key routing with a norm bonus and a NoPE long-range readout alongside local RoPE. FS-L prunes independently per head before query matching, bounding both routing and readout. MoBA and NSA process a growing set of summaries at fixed block size.

Sparse token retrieval. Landmark Attention (Mohtashami & Jaggi, 2023) learns landmark tokens to retrieve and weight blocks; its landmark cache grows with history. FS instead uses mean keys, and FS-L bounds the retained prefix. InfLLM retrieves blocks using representative tokens (Xiao et al., 2024a). Grouped Cross-Attention in DRT (Hu et al., 2025b) and Hierarchical Sparse Attention in RAMba (Hu et al., 2025a) learn distant chunk retrieval and extrapolate beyond training length. Their routing scores weight chunk outputs and receive gradients. Both offload growing memory to CPU. FS-L reads selected tokens with a joint softmax (Appendix D.2).

KV-cache selection and pruning. QUEST (Tang et al., 2024) uses coordinatewise key bounds to select pages for each query without persistent pruning. H_2O (Zhang et al., 2023) evicts tokens using accumulated attention and recency; SnapKV (Li et al., 2024) uses attention from a prompt observation window to select cached positions per head. Both estimate future access from observed queries. FS-L trains with pruning by maximum key norm and recency. Devoto et al. (2024) also prune by key norm, but retain low-norm keys in pretrained Transformers. FS-L favors high-norm cells and adapts the model to that rule; norm alone does not imply relevance.

NAtS-L routes chunks to retained-token softmax attention or GDN (Deng et al., 2026); its softmax history can grow. StreamingLLM keeps sinks and recent tokens (Xiao et al., 2024b); FS-L retains cells from anywhere in the prefix.

Sparsity and length generalization. ASEntmax links sparse weights to reduced dispersion and length extrapolation (Vasylenko et al., 2026); FS uses softmax on selected tokens. Position interpolation and YaRN rescale rotary positions (Chen et al., 2023; Peng et al., 2024); FS bounds their range. RoPE/NoPE hybrids (Yang et al., 2025a; Puvvada et al., 2025) combine global NoPE and local RoPE layers; SWAN-GPT also demonstrates length extrapolation with inference-time score scaling. FreeSolo instead selects token cells within each layer, and FS-L bounds the memory available for selection. Kimi Linear and Kimi K3 combine position-aware recurrence with global NoPE attention (Kimi Team, 2025; 2026). Both use long-context training.

7 Limitations

Our main limitation is scale: computational resources restrict experiments to 800M parameters. Larger models are needed to assess how retrieval and learned retention scale with language-model capability. A second limitation is FS-L's fixed retention budget. Because pruning precedes the query, it can discard details that later become relevant. The budget therefore trades memory use against coverage of possible questions; applications requiring many independently queryable facts may benefit from larger or adaptive budgets. We also explore FS-B, a balanced tree search targeting $O(N \log N)$ cost for information-dense retrieval (Appendix E). Streaming tree maintenance and broader evaluation remain future work.

8 Conclusion

We introduced FreeSolo, a native sparse attention mechanism for zero-shot length extrapolation. Its key idea is to separate local sequence modeling from distant evidence retrieval: RoPE operates within a short window, while keypoint selection identifies cells for direct NoPE attention. FS-L adds query-independent retention, learned through continued pretraining, to achieve linear attention cost and bounded state. Both variants train at 4k and retrieve single needles through 256k without longer-context training; FS-L also outperforms standalone Raven on harder retrieval tasks at extended lengths. Future work will explore adaptive memory budgets, test larger models, and develop custom kernels to improve throughput. We will open-source the code (Appendix F).

REFERENCES

Arshia Afzal, Aviv Bick, Eric P. Xing, Volkan Cevher, and Albert Gu. Raven: High-recall sequence modeling with sparse memory routing, 2026. URL https://arxiv.org/abs/2607.25357.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*, 2023.

Ta-Chung Chi, Ting-Han Fan, Alexander I. Rudnicky, and Peter J. Ramadge. Dissecting transformer length extrapolation via the lens of receptive field analysis. In *Proceedings of the Association for Computational Linguistics (ACL)*, 2023.

David Chiang and Peter Cholak. Overcoming a theoretical limitation of self-attention. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7654–7664. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.acl-long.527. URL https://aclanthology.org/2022.acl-long.527/.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 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. *arXiv preprint arXiv:2405.21060*, 2024.

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

Difan Deng, Andreas Bentzen Winje, Lukas Fehring, and Marius Lindauer. Neural attention search linear: Towards adaptive token-level hybrid attention models, 2026. URL https://arxiv.org/abs/2602.03681.

Alessio Devoto, Yu Zhao, Simone Scardapane, and Pasquale Minervini. A simple and effective L_2 norm-based strategy for KV cache compression, 2024. URL https://arxiv.org/abs/2406.11430.

Jusen Du, Weigao Sun, Disen Lan, Jiaxi Hu, and Yu Cheng. MoM: Linear sequence modeling with mixture-of-memories. *arXiv preprint arXiv:2502.13685*, 2025. URL https://arxiv.org/abs/2502.13685.

Lizhe Fang, Yifei Wang, Zhaoyang Liu, Chenheng Zhang, Stefanie Jegelka, Jinyang Gao, Bolin Ding, and Yisen Wang. What is wrong with perplexity for long-context language modeling? *arXiv preprint arXiv:2410.23771*, 2024.

Yao Fu, Rameswar Panda, Xinyao Niu, Xiang Yue, Hannaneh Hajishirzi, Yoon Kim, and Hao Peng. Data engineering for scaling language models to 128k context. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 14125–14134, 2024. URL https://proceedings.mlr.press/v235/fu24d.html.

Leo Gao et al. The language model evaluation harness, 2024. URL https://zenodo.org/records/12608602.

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

Zifan He, Yingqi Cao, Zongyue Qin, Neha Prakriya, Yizhou Sun, and Jason Cong. HMT: Hierarchical memory transformer for efficient long context language processing. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8068–8089. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.naacl-long.410. URL https://aclanthology.org/2025.naacl-long.410/.

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

Xiang Hu, Jiaqi Leng, Jun Zhao, Kewei Tu, and Wei Wu. Hardware-aligned hierarchical sparse attention for efficient long-term memory access. *arXiv preprint arXiv:2504.16795*, 2025a. URL https://arxiv.org/abs/2504.16795.

Xiang Hu, Zhihao Teng, Jun Zhao, Wei Wu, and Kewei Tu. Efficient length-generalizable attention via causal retrieval for long-context language modeling. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 24727–24743. PMLR, 2025b. URL https://proceedings.mlr.press/v267/hu25v.html.

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

Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. The impact of positional encoding on length generalization in transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Kimi Team. Kimi linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025.

Kimi Team. Kimi k3: Open frontier intelligence, 2026.

Andrew K. Lampinen, Stephanie C. Y. Chan, Andrea Banino, and Felix Hill. Towards mental time travel: A hierarchical memory for reinforcement learning agents. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://arxiv.org/abs/2105.14039.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation, 2024. URL https://arxiv.org/abs/2404.14469.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1711.05101.

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, et al. MoBA: Mixture of block attention for long-context LLMs. *arXiv preprint arXiv:2502.13189*, 2025.

Lovish Madaan, Srinadh Bhojanapalli, Himanshu Jain, and Prateek Jain. Treeformer: Dense gradient trees for efficient attention computation. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2208.09015.

Amirkeivan Mohtashami and Martin Jaggi. Random-access infinite context length for transformers. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2305.16300.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. *arXiv preprint arXiv:2406.17557*, 2024.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient context window extension of large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Ofir Press, Noah A. Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations (ICLR)*, 2022.

Krishna C. Puvvada, Faisal Ladhak, Santiago Akle Serrano, Cheng-Ping Hsieh, Shantanu Acharya, Somshubra Majumdar, Fei Jia, Samuel Kriman, Simeng Sun, Dima Rekesh, and Boris Ginsburg. SWAN-GPT: An efficient and scalable approach for long-context language modeling, 2025. URL https://arxiv.org/abs/2504.08719.

Noam Shazeer. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020. URL https://arxiv.org/abs/2002.05202.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017. URL https://arxiv.org/abs/1701.06538.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. QUEST: Query-aware sparsity for efficient long-context LLM inference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 47901–47911, 2024. URL https://proceedings.mlr.press/v235/tang24l.html.

Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000. URL https://arxiv.org/abs/physics/0004057.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

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

Pavlo Vasylenko, Hugo Pitorro, André F. T. Martins, and Marcos Treviso. Long-context generalization with sparse attention. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2506.16640.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992. doi: 10.1007/BF00992696.

Yuhuai Wu, Markus N. Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2203.08913.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. InfLLM: Training-free long-context extrapolation for LLMs with an efficient context memory. In *Advances in Neural Information Processing Systems*, 2024a. URL https://arxiv.org/abs/2402.04617.

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

Bowen Yang, Bharat Venkitesh, Dwarak Talupuru, Hangyu Lin, David Cairuz, Phil Blunsom, and Acyr Locatelli. RoPE to NoPE and back again: A new hybrid attention strategy, 2025a. URL https://arxiv.org/abs/2501.18795.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *International Conference on Learning Representations*, 2025b.

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

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Yu Zhang, Songlin Yang, Ruijie Zhu, Yue Zhang, Leyang Cui, Yiqiao Wang, Bolun Wang, Freda Shi, Bailin Wang, Wei Bi, Peng Zhou, and Guohong Fu. Gated slot attention for efficient linear-time sequence modeling. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2409.07146.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H_2O: Heavy-hitter oracle for efficient generative inference of large language models, 2023. URL https://arxiv.org/abs/2306.14048.

A HYPERPARAMETERS AND EXPERIMENTAL SETTINGS

Architecture. FreeSolo uses LLaMA-style decoders with RMSNorm, SwiGLU, and a 32k LLaMA tokenizer. Both scales have 24 layers and 16 heads. The 341M model has width 1024 and head width 64; the 743M model has width 1536 and head width 96. Attention uses $w = 512$, $B = 128$, $s = 32$, $K = 4$, $\sigma_h = 4$, and $\lambda_h = 1$. FS-L retains $P = 64$ norm-ranked cells and $W = 8$ recent cells, with 16 query blocks processed per kernel chunk.

Pretraining. We train at 4096 tokens in bf16 with AdamW, cosine learning-rate decay, weight decay 0.1, gradient clipping at 1.0, and 524,288 tokens per step. The 341M FS run and FS-L from-scratch control use peak learning rate 3×10^{-4}, floor 3×10^{-5}, and 1000 warmup steps. The 743M FS run uses 4×10^{-4}, 4×10^{-5}, and 2000 warmup steps. The 341M budget is 15B tokens (28,610 steps) over a 6.7B-token FineWeb-Edu subset, about 2.2 passes; the 743M budget is 32B tokens in less than one pass over a 39B-token corpus.

Continued pretraining. FS-L starts from FS with a fresh AdamW optimizer and pruning enabled. The learning rate warms up to 4×10^{-4} and decays toward 4×10^{-5}. The nominal 341M schedule has 150 warmup steps and 1500 total steps; the 743M schedule has 300 warmup steps and 3000 total steps. We use the last checkpoint from each reported run, at steps 1400 and 3000, corresponding to 734,003,200 and 1,572,864,000 additional tokens. Both use the same 4096-token context, tokenizer, and corpus as FS; restarting the data order revisits pretraining tokens.

RULER evaluation. We use NVIDIA RULER at commit `c3f5e3b4`, the LLaMA tokenizer, `base` prompt template, seed 42, and 100 examples per task and length. Generation caps are 128 tokens for NIAH, 30 for variable tracking, 120 for common-word extraction, 50 for frequent-word extraction, and 32 for QA. Decoding is greedy with EOS stopping. Non-QA tasks use `string_match_all` and QA uses `string_match_part`, on a 0–100 scale. MoBA and NSA suite summaries use a 32-token cap and `string_match_all` throughout. Lengths 2k through 64k denote 2048 through 65,536 tokens; table labels 131k and 256k denote 131,072 and 262,144 tokens.

Downstream evaluation. The eight zero-shot tasks are HellaSwag, PIQA, ARC-easy, ARC-challenge, WinoGrande, OpenBookQA, SciQ, and BoolQ. We use the LM Evaluation Harness with `acc_norm` for HellaSwag, ARC-challenge, and OpenBookQA, and `acc` otherwise.

Baselines. At 340M, NAtS-L uses 21 layers and 64-token routing blocks; both it and Raven are evaluated at step 28,610, or 15B tokens at 524,288 tokens per step. TF++ also uses a 15B-token checkpoint at this scale. Raven is standalone at both scales, with 21 layers, four heads, 256 memory slots, top-32 writes, and `gate_logit_normalizer=8`; its 726M model has width 1536. TF++ uses full RoPE attention, SWA a 512-token RoPE window, and InfLLM the TF++ checkpoint. The 800M GDN evaluations use the same checkpoint with strict weight loading: RULER uses `fla` 0.5.2 and `transformers` 5.14.1; downstream tasks use versions 0.3.2 and 4.56.1, respectively, with LM Evaluation Harness 0.4.13.

B FULL RULER TASK BREAKDOWN

Table 3 reports individual task scores. FS and FS-L improve long-context retrieval, while multi-key retrieval, variable tracking, and word aggregation remain difficult. Raven retains strong S-NIAH-1 recall at 800M, while FS-L performs better on S-NIAH-2/3 and the suite mean. FS-L uses CPT at both scales.

MK, MQ, and MV denote multi-key, multi-query, and multi-value NIAH; VT denotes variable tracking; CWE and FWE denote common- and frequent-word extraction. QA-2 is unavailable at 2k/4k, so the mean covers 12 tasks there and 13 at 8k/16k. Best and second-best scores are compared across models for each task and length within each complexity group and scale, including ties. NAtS-L's linear-time grouping assumes a bounded retained-token count (Section 4.1).

Table 3: **RULER results by task** (0–100) at 340M and 800M. Bold and underline mark the best and second-best scores within each complexity group and scale.

	$O(N^2)$								$O(N)$											
	TF++				**FS**				**FS-L**				NAtS-L				Raven			
Task	2k	4k	8k	16k	2k	4k	8k	16k	2k	4k	8k	16k	2k	4k	8k	16k	2k	4k	8k	16k
≈340M scale																				
S-NIAH-1	95	57	0	0	**99**	**99**	**100**	**100**	99	87	89	**87**	**100**	94	37	19	95	**96**	**93**	81
S-NIAH-2	91	24	0	0	**100**	**77**	**41**	**17**	**99**	**72**	**39**	**18**	89	22	20	3	92	70	36	3
S-NIAH-3	38	1	0	**0**	**58**	**14**	**7**	**0**	**70**	**16**	**8**	**1**	36	6	1	0	3	0	0	0
MK-NIAH-1	**27**	**20**	0	0	**27**	**20**	**15**	**5**	**30**	**22**	**18**	**10**	17	18	10	4	11	11	10	3
MK-NIAH-2	**1**	**2**	**0**	**0**	**1**	0	**0**	**0**	**0**	**1**	**0**	**0**	**0**	0	**0**	**0**	**0**	0	**0**	**0**
MK-NIAH-3	**1**	**0**	**0**	0	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
MQ-NIAH	20.0	14.8	0.0	0.0	**20.2**	**20.8**	**12.8**	**2.8**	**21.2**	**21.0**	10.5	**3.8**	17.5	15.2	4.8	2.8	15.0	16.5	**13.2**	3.0
MV-NIAH	15.8	14.0	0.0	0.0	**23.2**	**18.5**	**15.5**	**9.8**	**24.2**	**20.0**	**16.0**	**7.2**	18.8	8.8	4.5	7.0	11.2	12.0	12.0	4.5
VT	**3.2**	0.0	0.0	0.0	2.2	**1.2**	**0.6**	**1.6**	2.0	0.0	0.0	0.0	**4.8**	0.6	0.0	0.0	2.6	**2.0**	**1.4**	**1.0**
CWE	2.2	**14.8**	0.1	0.0	**4.1**	9.2	**5.2**	**0.8**	**5.5**	**9.0**	**3.8**	**0.4**	2.6	5.0	0.2	0.3	1.0	0.2	0.2	0.1
FWE	1.3	0.3	0.0	0.0	**8.7**	**3.3**	**5.0**	**3.3**	**10.0**	0.7	**3.0**	**3.3**	3.0	**1.7**	0.3	2.3	0.0	0.0	0.0	0.0
QA-1	19	12	1	0	**25**	**15**	**5**	**7**	**26**	8	4	3	19	**12**	**9**	4	15	6	4	**6**
QA-2	–	–	1	0	–	–	**8**	**5**	–	–	10	5	–	–	10	**10**	–	–	**12**	9
Available-task mean	26.2	13.3	0.2	0.0	**30.8**	**23.2**	**16.5**	**11.8**	**32.3**	**21.4**	**15.5**	**10.7**	25.6	15.3	7.4	4.0	20.5	17.8	14.0	8.5
≈800M scale																				
S-NIAH-1	**100**	**95**	0	0	97	91	**93**	**95**	**100**	93	83	84	86	72	59	21	**100**	**100**	**100**	**100**
S-NIAH-2	**100**	94	0	0	**100**	**99**	**92**	**72**	99	**99**	**89**	**81**	**100**	76	28	3	99	97	81	26
S-NIAH-3	67	50	0	0	**85**	**71**	**51**	**34**	**91**	**72**	**55**	**38**	62	24	6	0	32	7	1	0
MK-NIAH-1	**79**	**70**	0	0	53	32	**30**	**28**	**42**	**32**	**27**	**24**	31	20	7	3	30	19	14	3
MK-NIAH-2	**18**	**11**	0	0	7	0	**1**	**1**	**8**	**3**	**1**	**1**	0	0	0	0	3	0	0	0
MK-NIAH-3	**8**	**0**	**0**	**0**	0	**0**	**0**	**0**	**2**	**0**	**1**	**0**	0	**0**	0	**0**	0	**0**	0	**0**
MQ-NIAH	**56.5**	**38.5**	0.0	0.0	34.5	25.8	**22.8**	**23.0**	25.8	**22.5**	**16.2**	**15.5**	16.5	11.5	1.5	0.3	**27.8**	19.8	11.0	7.2
MV-NIAH	**56.0**	**33.3**	0.0	0.0	28.0	26.3	**22.8**	**18.8**	**27.5**	**21.0**	**21.5**	**16.0**	13.5	10.8	3.0	0.5	25.8	17.0	13.5	9.0
VT	0.4	**2.8**	0.0	**0.0**	**1.6**	0.2	**0.4**	**0.0**	**3.8**	**0.8**	**0.6**	0.6	0.0	0.2	0.0	0.0	1.6	**0.8**	**0.6**	**1.0**
CWE	**13.7**	**28.4**	0.0	0.0	8.4	26.7	**13.5**	**3.3**	**5.0**	**12.5**	**8.2**	**1.1**	0.4	0.7	0.0	0.0	1.9	1.3	0.1	0.1
FWE	**17.3**	2.7	0.0	0.0	16.7	**6.0**	**5.0**	**5.7**	0.3	0.0	1.7	0.0	1.7	3.3	**4.7**	0.0	**18.0**	**11.0**	1.3	**3.7**
QA-1	40	24	0	0	**42**	**30**	**12**	**9**	**43**	**28**	10	**11**	23	14	**11**	0	22	14	6	5
QA-2	–	–	0	0	–	–	**15**	**11**	–	–	**19**	**16**	–	–	12	9	–	–	11	11
Available-task mean	**46.3**	**37.5**	0.0	0.0	39.4	34.0	**27.6**	**23.1**	**37.3**	**32.0**	**25.6**	**22.2**	27.8	19.4	10.2	2.8	30.1	23.9	18.4	12.8

C EFFICIENCY BENCHMARKS

The streaming implementation maintains fixed attention buffers and nearly constant decode throughput from 4k to 256k. Absolute throughput still lags optimized local and recurrent baselines. We measure training, prefill, and decoding separately. These benchmarks use the models' available optimized kernels; the results depend on batch size, software stack, and execution mode as well as architecture.

Training. Table 4 reports aggregate forward-and-backward throughput on eight A100-40GB GPUs, with 4096-token sequences, bf16, and `torch.compile`. All rows use batch 1 per GPU. Raven and NAtS-L are measured with the fla/flame (FSDP) harness (sharded parameters and optimizer states), so their peak memory is not directly comparable and is omitted, and their throughput carries some FSDP communication overhead. Bold and underline mark the best and second-best results (higher throughput, lower memory).

GDN uses flash-linear-attention, NSA uses its Triton kernel, MoBA uses SDPA, and FreeSolo uses fused indexed-cell attention. FS, TF++, and SWA run on PyTorch 2.5; the other models use PyTorch 2.9. TF++ is fastest at 303K tokens/s. FS and FS-L reach 99.3K and 98.4K tokens/s, with peak memory of 16.5 and 16.6 GiB. Sparse long-range retrieval reduces scaling cost, but has overhead at the short training context.

Inference. Table 5 uses one A100-40GB, bf16, and batch 1. Prefill is one forward pass over the prompt; decode processes one new token using each model's KV cache or recurrent state. All table entries use eager execution, without `torch.compile` or CUDA-graph capture; bold and underline mark the fastest and second-fastest results in each column. TF++ uses FlashAttention-2 (Dao, 2023); FreeSolo uses fused indexed-cell prefill and a fused sliding-window local branch. The other baselines use their native attention or recurrent kernels, with an incremental reference path for NSA decoding.

FS-L prefill remains between 34.6K and 44.2K tokens/s over the measured 4k–131k range. At 131k it reaches 39.6K, compared with TF++'s 26.2K; FS runs out of memory. FS-L's eager decode stays near

Table 4: **Training throughput and peak memory** at 340M scale on eight A100-40GB GPUs. Raven and NAtS-L use the fla/flame FSDP harness; see the text.

Model	Train tok/s	Peak mem (GiB)
TF++ (full attention)	**303K**	**15.0**
SWA (sliding-512)	228K	15.5
GDN	158K	21.6
MoBA	140K	27.0
NSA	28K	31.9
Raven	140K	–
NAtS-L	126K	–
FS	99.3K	16.5
FS-L	98.4K	16.6

20 tokens/s. Several baselines are faster, including SWA at about 33 tokens/s. The measured benefit is sustained long-context retrieval with bounded query scoring, rather than the highest throughput at every length. These measurements use the full-history reference, including its KV storage and prefix priority scans (Section 3.4). A table entry is marked OOM when peak memory exceeds the 40 GB budget: FS fails at 131k for both prefill and decode, and MoBA for decode.

Table 5: **Prefill and eager decode throughput** at 340M scale on one A100-40GB (bf16, batch 1). OOM denotes the 40 GB memory limit.

	Prefill (Ktok/s)				Decode (tok/s)			
Model	4k	16k	64k	131k	4k	16k	64k	131k
TF++ (full attention)	**144.4**	123.8	48.8	26.2	**34.7**	**34.7**	**34.7**	20.5
SWA (sliding-512)	143.5	**218.8**	**248.6**	**245.3**	33.1	33.2	33.2	**33.3**
MoBA	34.3	48.6	33.3	20.2	27.4	26.9	26.1	OOM
NSA	87.0	121.4	89.8	65.2	15.8	15.7	15.8	15.8
GDN	65.7	141.8	146.8	146.2	25.5	25.7	25.6	25.4
NAtS-L	56.1	132.6	107.6	83.7	22.5	22.7	22.2	22.9
Raven	71.3	137.7	147.6	148.9	24.7	24.8	24.8	24.8
FS	30.9	22.3	10.9	OOM	19.0	18.7	18.2	OOM
FS-L	34.6	43.6	44.2	39.6	19.8	19.9	19.8	19.8

Compilation and CUDA graphs. With `torch.compile`, FS-L prefill reaches 62.8/66.8/68.2/67.6K tokens/s at 4k/16k/64k/131k. FS reaches 46.6/35.6/21.4K through 64k and still runs out of memory at 131k. Compilation mainly improves prefill; it does not improve eager decode in this measurement. These results are reported separately from the eager comparison in Table 5.

CUDA-graph capture raises FS-L decode throughput to 94.7/94.5/94.4 tokens/s at 4k/16k/64k, with peak memory of 1.9/5.5/19.6 GiB. The full-history graph cache runs out of memory at 131k on the A100-40GB. This optimization also benefits the smaller recurrent and windowed computations: SWA, GDN, and Raven reach approximately 380, 275, and 262 tokens/s over this range. These additional measurements show the effect of execution overhead; Table 5 uses eager decode throughout for a common comparison.

Bounded cache. The streaming implementation updates top-P from newly eligible cells and physically evicts unused KV. It keeps absolute cell indices for relative scores and computes RoPE locally. Prefill processes the transformer in chunks of 2048 tokens, with retention updates every $B = 128$ tokens. Decode uses a fixed local ring and compact cell buffers; pool updates occur outside the CUDA graph and are included in the timing.

Table 6 uses both CPT checkpoints, bf16 parameters and activations, and FP32 rotary frequencies on one A100-40GB. We report the median of three runs after one warmup, with 257 decode steps per

run. Attention-buffer bytes count distinct physical allocations across all heads and layers, including retained, recent, local, pending, and graph input buffers. The implementation keeps duplicate compact buffers, so this exceeds the size of one packed retained set. Peak allocated GPU memory also includes parameters, activations, and graph workspace; input/output token IDs are excluded from the attention-buffer count.

Table 6: **Bounded FS-L inference.** Throughput uses CUDA-graph decode; peak memory is the maximum across three trials.

Length	Attention buffers (MiB)	Peak (GiB)	Prefill (Ktok/s)	Decode (tok/s)
340*M scale*				
4k	766.2	1.71	2.20	89.3
64k	766.2	1.74	2.21	89.4
256k	766.2	1.77	2.17	88.3
800*M scale*				
4k	1148.8	2.96	2.09	75.2
64k	1148.8	2.98	2.14	75.1
256k	1148.8	3.02	2.14	74.1

This establishes bounded attention storage with linear streaming work, at a substantial prefill cost relative to Table 5. Both implementations apply the same pruning rule, but floating-point precision and reduction order can change generated answers. The reference uses FP32 parameters with bf16 autocast; CUDA-graph measurements use bf16 parameters and activations. On 25 prompts per S-NIAH task at 4k and 16k, bounded eager execution at 340M matches all 150 reference correctness decisions, with 123 identical continuations. The measured bf16 graph configuration matches $60/60$ decisions at 340M and $59/60$ at 800M on ten prompts per task and length; one 800M S-NIAH-1 answer changes from correct to incorrect. The adapted $P = 16$ and $P = 32$ checkpoints each match $58/60$ decisions under the same graph check. At $P = 16$, one answer improves and one worsens; at $P = 32$, two answers improve. Both scales retrieve all three tested 256k S-NIAH-1 examples. These checks cover a subset of the benchmark; the accuracy tables use reference evaluation.

D Theory and ablation studies

We examine position handling, retrieval through cell summaries, and pruning before future queries. The ablations and diagnostics support the discussion in Section 5; the final subsection gives the incremental update that realizes linear scaling.

D.1 Local position modeling and the long-range norm prior

The ablations separate short-context quality from length extrapolation: changing the local branch sharply affects downstream accuracy, while distant retrieval persists. In long-range routing, the erf term acts mainly as a norm bonus under the reported hyperparameters.

Position dependence. Local scores use $\mathbf{q}_i^\top R(j-i)\mathbf{k}_j/\sqrt{d}$ with $0 \le i - j \le w$. Training at length $L > w$ covers every explicit rotary offset used at evaluation. Long-range token scores are NoPE. The cell prior $g_h(i - p_c)$ is unchanged by shifting query and cell coordinates together, and approaches one with distance. At fixed content vectors and a consistent tie rule, translating tokens, cell partitions, query blocks, and causal boundaries together leaves the operator unchanged. These properties avoid new rotary offsets; they do not make contextual representations invariant to added tokens.

Trained ablations. We train both controls with the 341M FS recipe, 4k context, and 15B training tokens. SR-NoPE replaces local RoPE with unrotated queries and keys and adds a log-$\mathrm{erfc}(\Delta/2\sigma_h)$ bias, retaining the 512-token window. It therefore changes both rotation and local bias. LR-no-erf keeps local RoPE and removes the entire $\lambda_h a_c \,\mathrm{erf}(\Delta/2\sigma_h)$ routing term, leaving query–mean-key similarity alone.

Table 7: **RULER ablations at** 340**M** (0–100). Bold and underline mark the best and second-best scores, including ties.

	S-NIAH-1							S-NIAH-2				S-NIAH-3			
Model	2k	4k	8k	16k	32k	64k	131k	2k	4k	8k	16k	2k	4k	8k	16k
FS	99	99	**100**	**100**	98	**100**	98	**100**	77	41	17	58	14	7	0
SR-NoPE	**100**	**100**	**100**	**100**	**100**	**100**	**100**	98	**90**	**71**	**29**	47	23	**13**	0
LR-no-erf	95	97	96	97	95	95	96	80	57	38	14	**71**	**25**	11	**1**

Table 8: **Downstream accuracy of the ablations** (0–100), using the metrics in Table 2.

Model	HellaS.	PIQA	ARC-e	ARC-c	WinoG.	OBQA	SciQ	BoolQ	Avg
FS	38.9	**66.8**	**57.6**	25.9	50.9	**32.4**	81.2	**61.1**	51.9
SR-NoPE	28.4	55.7	36.9	23.7	49.2	25.6	48.2	38.7	38.3
LR-no-erf	**39.3**	66.1	57.1	**26.9**	**52.8**	31.8	**83.2**	60.0	**52.1**

SR-NoPE preserves S-NIAH-1 extrapolation and improves S-NIAH-2, but lowers average downstream accuracy from 51.9 to 38.3. The local FS configuration therefore improves language-model quality without being required for single-needle retrieval. LR-no-erf retains 95–97 on S-NIAH-1 and a downstream average of 52.1, while worsening S-NIAH-2 and improving S-NIAH-3. Its effects are task dependent and cannot be attributed to the erf shape alone.

A saturated norm prior. Every eligible cell satisfies $(c+1)s \le b_i \le i$, so its center $p_c = (c+\frac{1}{2})s$ obeys $i - p_c \ge s/2$. Write the content term as $u_{ic} = \mathbf{q}_i^\top \bar{\mathbf{k}}_c/\sqrt{d}$. For $\lambda_h \ge 0$, define the constant-gate score $r_{ic}^{\infty} = u_{ic} + \lambda_h a_c$. Then

$$0 \le r_{ic}^{\infty} - r_{ic} = \lambda_h a_c \operatorname{erfc}\left(\frac{i-p_c}{2\sigma_h}\right) \le \lambda_h a_c \operatorname{erfc}\left(\frac{s}{4\sigma_h}\right). \tag{11}$$

At $s = 32, \sigma_h = 4$, $\operatorname{erfc}(2) \simeq 0.004678$: the difference is at most 0.468% of the norm bonus. This is not a relative bound on the total score, which can include cancellation with the content term. Larger offsets further reduce the difference.

For a fixed available set with more than K cells, let γ_∞ be the gap between the Kth and $(K+1)$st constant-gate scores and $\delta_{\max} = \max_c \lambda_h a_c \operatorname{erfc}((i-p_c)/(2\sigma_h))$. If $\gamma_\infty > \delta_{\max}$, restoring erf leaves the top-K set unchanged: a selected score falls by at most $\delta_{\max}$, while an unselected score cannot increase. The raw-token readout is then unchanged too. This gives a testable prediction: routing changes require small boundary margins relative to the gate correction. It applies to FS's full prefix and FS-L's retained set at fixed representations.

Table 9: Routing controls separate the norm bonus from its distance dependence.

Routing control	Added term	What it tests
Erf gate	$\lambda_h a_c\, g_h(i-p_c)$	Norm bonus with distance gate
Constant gate $g_h \equiv 1$	$\lambda_h a_c$	Remaining distance dependence
LR-no-erf (content-only)	0	Entire norm-weighted bonus

Fixed-state routing controls. On the 16k native traces described in Appendix D.3, replacing g_h by one preserves 99.998% of selected-cell memberships in both 340M models. Removing the entire bonus instead preserves only $53.9/14.0/14.7\%$ in FS and $73.3/32.5/32.7\%$ in FS-L on S-NIAH-1/2/3. The norm bonus changes routing substantially, whereas its residual distance dependence rarely changes the selected set. A separately trained constant-gate control is still needed to isolate that distance dependence after learning.

D.2 CELL SUMMARIES RESTRICT SELECTION, NOT READING DISTANCE

FS imposes a bottleneck on cell selection: a query selects cells using mean keys and maximum key norms, then reads the selected cells' original keys and values. The routing summary need only locate useful evidence; it need not contain the answer. Sparse readout also bounds how many token values can interfere, independently of their distance from the query.

A keypoint can mark a cell. If a cell contains one nonzero key $\mathbf{k}_\star$, its mean is $\mathbf{k}_\star/s$, while $a_c = \|\mathbf{k}_\star\|_2$. The content match is diluted by $1/s$, but the norm bonus preserves the cue's magnitude. This illustrates how one key can make a cell visible to the router, but does not establish that large-norm keys carry useful information. A high-norm distractor can also receive a large bonus; query alignment and the learned representations determine which cells are useful.

Why sparse readout matters after RMSNorm. Let $\mathbf{z} = \sum_{j\in\mathcal{S}_i} e^{\ell_{ij}}\mathbf{v}_j$, $Z = \sum_{j\in\mathcal{S}_i} e^{\ell_{ij}}$, and $\boldsymbol{\gamma}$ be the readout RMSNorm gain. Then

$$\mathbf{o}_i^{\mathrm{lr}} = \frac{\boldsymbol{\gamma}\odot\mathbf{z}}{\sqrt{\|\mathbf{z}\|_2^2/d + \epsilon Z^2}}. \tag{12}$$

When $\|\mathbf{z}/Z\|_2^2/d \gg \epsilon$ ($\epsilon = 10^{-8}$), the scalar softmax denominator cancels. Normalization therefore removes amplitude dilution, but irrelevant values can still alter the output direction. If the value sum nearly cancels, the stabilizer can also matter. Sparse selection restricts which values enter this sum, rather than merely reducing its softmax denominator. This distinction matters when relating FreeSolo to analyses of attention dilution and sparse length generalization (Chiang & Cholak, 2022; Vasylenko et al., 2026).

Read distance. Raven measures memory through the number of tokens written to recurrent slots (Afzal et al., 2026, Definition 4 and Figure 8). For FreeSolo's fixed cell contents, we trace where queries read. Selecting cell c at query i gives backward distance $i - p_c$ tokens, where $p_c = (c + \frac{1}{2})s$ and $s = 32$. Figures 3 and 4 plot this distance against query position, without sorting. Both axes use tokens (1k=1024) and 2048-token bins. Color shows the fraction of cell selections per query bin, aggregated over the layer's 16 heads, on a logarithmic scale. White denotes zero selections; gray lies beyond the available prefix.

All four checkpoints receive the same 262,010-token S-NIAH-1 prompt at nominal 256k. FS-L uses 1400 CPT steps at 341M and 3000 at 743M. The needle occupies cell 2981, centered at $p_\star = 95{,}408$. The orange dotted line marks its position; the dashed diagonal follows its distance $i - p_\star$ from later queries. Diagonal bands indicate repeated access to fixed earlier cells.

All four checkpoints read the needle cell in layer 1. At the final prompt query, 166,601 tokens from its center, layer 24 selects it in $5/16$ FS heads and $1/16$ FS-L heads at 340M, and $6/16$ and $2/16$ at 800M. Over the final 2048 queries, averaging all layers and heads, 92.8% of FS selections and 90.4% of FS-L selections at 340M lie at least 131,072 tokens behind the query. The corresponding 800M fractions are 93.6% and 93.1%. These traces show direct access to distant cells; they do not measure attention mass or contribution to the answer.

Comparison with learned chunk retrieval. DRT uses Grouped Cross-Attention (GCA) to retrieve past chunks for the next chunk (Hu et al., 2025b). RAMba combines Mamba-2 with per-token Hierarchical Sparse Attention (HSA), sharing chunk selection across its attention layers (Hu et al., 2025a). Both preserve distant token access. Their routing scores also weight chunk outputs, giving the router a gradient through the readout. FreeSolo uses hard cell selection followed by one softmax over the selected tokens; its shared projections learn through token readout and local attention (Table 10).

DRT and RAMba offload token memory while keeping a growing set of chunk keys on the GPU. This reduces device memory without bounding total stored history. FS-L bounds accessible cells by $P + W$ per head; realizing the corresponding storage bound requires eviction (Section 3.4). The published downstream protocols also differ: DRT uses task fine-tuning, and RAMba's RULER-based tests include synthetic fine-tuning at 4k. Our reported models use next-token pretraining only. These differences preclude a matched numerical comparison with their published scores.

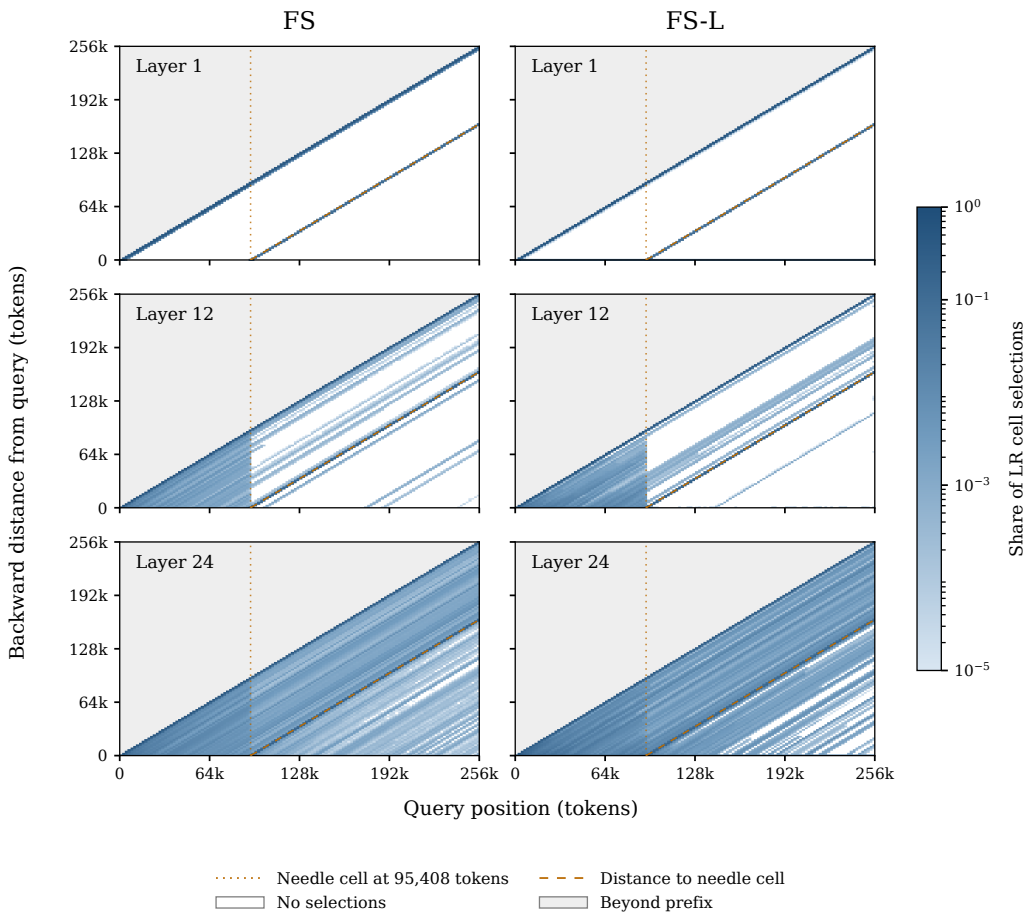

Figure 3: **Long-range read distance at** 340**M** on one 256k S-NIAH-1 prompt. Orange guides mark the needle position and its distance from each query.

Table 10: **Chunk retrieval and retained history.**

Model	Retrieval	Routing-score gradient	Stored history
DRT (GCA)	Chunk selects past chunks	Through softmax-weighted chunk outputs	Growing chunk memory; CPU offloading
RAMba (HSA)	Token selects past chunks	Through stick-breaking chunk weights	Growing token memory and selection keys; CPU offloading
FS	Query selects cells from full prefix	None through hard indices	Full token history
FS-L	Norm retention, then query selects cells	None through hard indices	Bounded cell buffers with streaming eviction

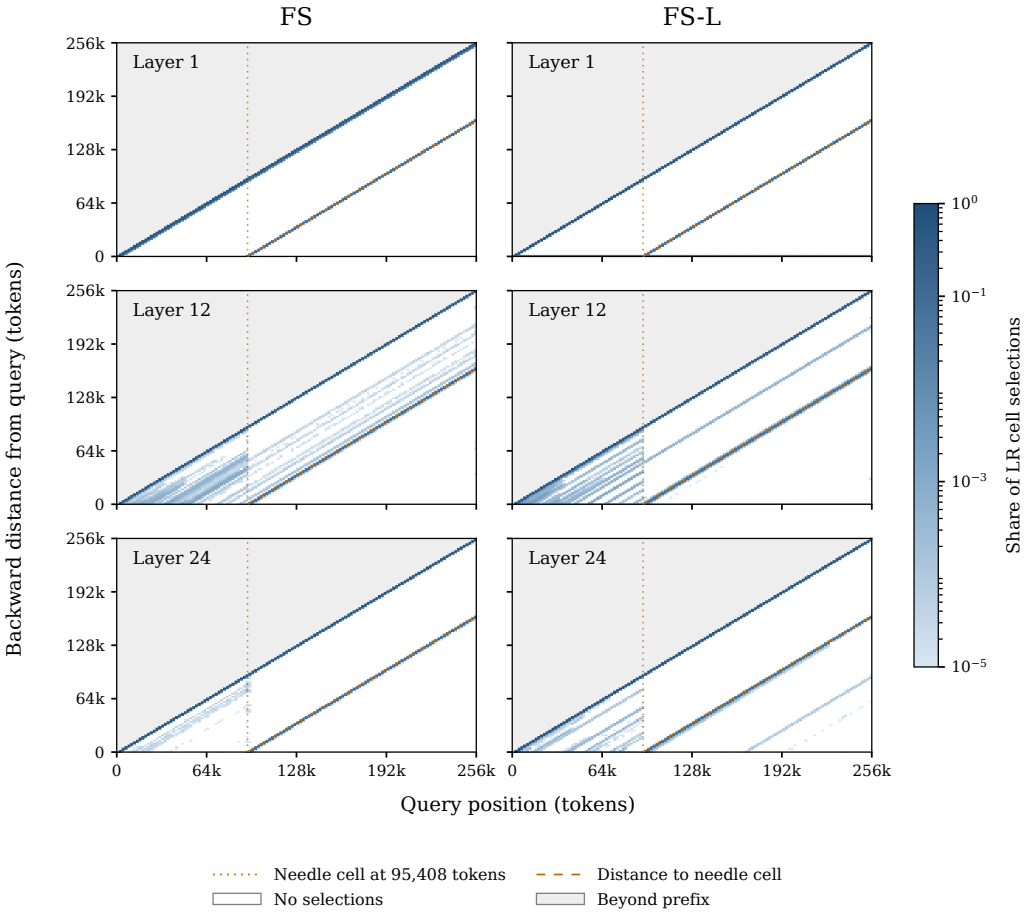

Figure 4: **Long-range read distance at** 800**M** on one 256k S-NIAH-1 prompt. Orange guides mark the needle position and its distance from each query.

D.3 NORM PRUNING AND RETRIEVAL COVERAGE

We measure how much norm pruning reduces the available prefix and how it affects retrieval. Key norm is a learned pruning heuristic: a smaller-norm cell can still be relevant to a future query. We therefore evaluate source access, selection coverage, and answer accuracy separately.

Separate retention from selection. A non-recent cell survives pruning when fewer than P cells outrank it by a_c. It is then read only if fewer than K retained cells outrank it by r_{ic}. Raising K cannot recover a cell already pruned. At fixed scores, retaining every FS top-K cell preserves its selected-token readout, but partial overlap need not preserve the answer. We measure these stages separately.

Probe protocol. We trace the reported FS and CPT FS-L checkpoints at both scales on 50 examples per S-NIAH task at 2k/4k/8k/16k, totaling 600 prompts per model. Each trace covers all 24 layers and 16 heads on the final 16 prompt positions. The forward pass uses each model's native routing with the fused kernel. Alternative scores are computed on those same representations without changing the forward pass. Source-access measurements exclude prompts whose answer source overlaps the local window of any observed query. We average head/query rates within each prompt, then average prompts; heads are not independent observations.

Source availability and selection. Table 11 distinguishes source retention from query-dependent selection. FS keeps the full prefix, so its retention column applies the FS-L rule offline to FS keys. Its selection column uses native FS routing. FS-L's columns both describe its native operation.

Table 11: **Source-cell availability and selection at 16k.**

		Source retained (%)		Source selected (%)	
Task	Prompts	FS (offline)	FS-L	FS	FS-L
340M scale					
S-NIAH-1	47	22.31	**27.98**	**38.15**	17.55
S-NIAH-2	50	18.34	**19.87**	1.27	**1.30**
S-NIAH-3	47	24.64	**25.76**	2.49	**2.65**
800M scale					
S-NIAH-1	47	**23.07**	22.03	**29.21**	14.99
S-NIAH-2	50	16.22	**17.03**	2.00	**2.08**
S-NIAH-3	47	26.29	**26.42**	**3.55**	3.52

At 340M, under the same norm rule, CPT FS-L improves source availability by $5.66/1.54/1.11$ percentage points on S-NIAH-1/2/3. Paired prompt bootstrap 95% intervals are $[5.05, 6.31]$, $[1.14, 1.93]$, and $[0.72, 1.49]$ (5000 resamples). These compare the two reported checkpoints; they do not isolate pruning adaptation from additional training. Native source selection is substantially lower in FS-L on S-NIAH-1 and similar on the harder tasks. Among head/query pairs where the source survives, FS-L selects it in $63.8/6.6/10.3\%$ of cases, averaging the conditional rate within each prompt. Averaged access rates do not measure whether any head retrieves the source, nor whether that access causes a correct answer. Joining the same eligible prompts to greedy answers gives $47/47$, $9/50$, and $0/47$ correct for FS, and $40/47$, $9/50$, and $0/47$ for FS-L on S-NIAH-1/2/3 at 16k. Source access therefore need not produce a correct answer. These joins compare fused probe traces with masked-pool answer evaluation, so they do not establish that a traced head caused the answer. Contextual states can also carry information from other cells.

At 800M, the corresponding source-availability changes are $-1.04/0.82/0.13$ points, with paired 95% intervals $[-2.06, 0.07]$, $[0.47, 1.16]$, and $[-0.35, 0.64]$. The increase at 340M therefore does not generalize uniformly across scales. Native S-NIAH-1 source selection is again lower in FS-L, while S-NIAH-2/3 selection changes little. FS-L selects a surviving source in $69.3/12.3/13.4\%$ of eligible head/query pairs, averaging within each prompt. These measurements support separate retention and selection diagnostics, without attributing CPT's answer quality to a universal increase in source retention.

Compression. For each head, define the retained fraction $\rho_i = |\mathcal{A}_i|/|\mathcal{C}_i|$. At 4k/8k/16k, native FS-L retains $58.8/29.0/14.3\%$ of eligible cells across these probes, corresponding to $1.7/3.4/7.0\times$ reductions in cells available for scoring. All lengths use $P = 64, W = 8$. These fractions concern cell counts; Appendix C measures physical storage.

Offline retention budgets. A separate probe evaluates $P \in \{16, 32, 64, 128\}$ on 25 examples per task and length, using the final 16 prompt positions. Both checkpoints run with unpruned routing to provide a common counterfactual: which of the full-prefix top-K cells would survive each norm budget? Thus these measurements do not trace native FS-L hidden states and do not report accuracy after changing P.

Table 12: **Offline pruning of unpruned routes at 16k (340M).**

		Memberships kept (%)		Full top-K kept (%)	
P	Cells kept (%)	FS	CPT FS-L	FS	CPT FS-L
16	4.79	74.99	**76.94**	50.98	**52.73**
32	7.95	81.53	**83.57**	63.98	**66.55**
64	14.28	85.31	**87.52**	71.90	**74.81**
128	26.94	87.67	**89.78**	76.56	**79.40**

Table 12 reports retained memberships and the fraction of head/query pairs retaining all four selected cells. Cell fractions use CPT representations; FS fractions differ by less than 0.02 percentage points. Increasing P improves routing agreement, but retaining high-priority cells still discards some cells that queries would select.

D.4 Continued pretraining adapts retrieval to pruning

CPT recovers much of the S-NIAH-1 accuracy lost under immediate pruning. Harder-task changes also occur with unpruned continuation and vary across seeds. Training with pruning from scratch is possible at 340M, but CPT improves overall RULER performance, particularly on harder retrieval tasks. The tested 800M from-scratch run retrieves poorly. Together, these results support adapting an FS checkpoint to the pruning constraint through short CPT.

Learning with hard indices. Top-P and top-K indices receive no ordinary loss gradient; we use no auxiliary retention loss or straight-through estimator. Shared query/key projections and contextual states learn through local attention and selected-token readout, so later forward passes can assign different scores and norm ranks. A key receives no gradient through an access that excludes it, and the remaining gradients need not increase its norm. Routing-only σ_h and λ_h stay fixed in this implementation (Section 3.3).

CPT starts from FS's retrieval representations and keeps pruning in the loop at the same 4096-token context. The constraint already binds during training: the last query block has 124 eligible cells, exceeding $P + W = 72$. Local and recent access provide learning signals before long-term retention is needed. Optimization settings are in Appendix A.

Continuation controls. We compare immediate pruning without further training, continuation without pruning, and continuation with pruning. Each continuation starts from the same base FS checkpoint and uses a fresh optimizer, the same corpus and schedule, and 1500 steps (0.786B tokens) at 340M. FS continues and evaluates with full-prefix routing; FS-L uses $P = 64$ during continuation and evaluation. Thus the trained arms compare quality under different native memory budgets. Immediate pruning and FS-L adaptation share a retention budget and test recovery after training.

Table 13 uses the same first 50 examples per task and length in every row, including the reported run's final 1400-step checkpoint. The additional continuations use their final 1500-step checkpoints. We run seed 42 twice with the same data order and seed 43 once; changing the seed also changes data sampling.

Table 13: **Continuation controls at** 340**M.** Shared 50 examples per task and length.

	S-NIAH-1				S-NIAH-2				S-NIAH-3			
Model	2k	4k	8k	16k	2k	4k	8k	16k	2k	4k	8k	16k
FS base	**100**	**100**	**100**	**100**	**100**	74	38	**14**	64	14	4	0
Immediate pruning	**100**	62	50	50	**100**	74	42	**14**	66	12	4	0
FS-L reported (1400)	**100**	90	92	86	98	66	30	12	62	12	6	**2**
FS, seed 42 (run 1)	**100**	**100**	98	96	96	50	24	4	58	16	**8**	**2**
FS, seed 42 (run 2)	**100**	**100**	96	96	94	52	30	4	**74**	**20**	6	**2**
FS, seed 43	**100**	**100**	**100**	98	98	74	46	8	60	6	2	0
FS-L, seed 42 (run 1)	**100**	86	78	76	94	46	18	2	62	12	**8**	**2**
FS-L, seed 42 (run 2)	**100**	78	72	72	94	48	22	2	60	18	**8**	**2**
FS-L, seed 43	**100**	82	72	60	**100**	**80**	**50**	**14**	62	6	2	0

At 4k, immediate pruning reduces S-NIAH-1 from 100 to 62. The two seed-42 adapted runs recover 86 and 78, gains of 24 and 16 percentage points (paired-prompt bootstrap 95% intervals $[10, 38]$ and $[0, 32]$). Both unpruned continuations retain 100. In contrast, immediate pruning preserves S-NIAH-2 at 74, while continuation reduces it to $50/52$ without pruning and $46/48$ with pruning. This harder-task decline also occurs without a retention bottleneck. At 64k, S-NIAH-1 is $100/98$ for FS continuation and $88/78$ for FS-L continuation; all four score $2/2$ on S-NIAH-2/3.

Seed 43 recovers S-NIAH-1 to 82 at 4k and changes the harder-task comparison: S-NIAH-2 is 80 with pruning and 74 without it, versus $46/48$ and $50/52$ in the seed-42 repeats. The reported FS-L checkpoint's 66 on this subset lies within the observed 46–80 range. These results support recovery of S-NIAH-1 under pruning, but not a consistent harder-task benefit from pruning. Intervals use 5000 prompt resamples and condition on the trained checkpoints; they do not estimate seed uncertainty and are not adjusted for multiple comparisons.

At 800M, we continue FS for the same 3000 steps (1.57B tokens) as the reported FS-L checkpoint, with pruning disabled. Table 14 compares the base checkpoint, unpruned continuation, and FS-L on identical prompts. At 16k, continuation improves S-NIAH-2/3 from $72/32$ to $84/38$; FS-L scores $80/42$ with bounded state. Its S-NIAH-3 advantage is 4–8 points across lengths, but the paired intervals do not establish a consistent gain. S-NIAH-1 accuracy is 10–16 points lower at 4k–16k than with unpruned continuation. The models use different inference budgets, so the comparison measures the tradeoff between quality and storage.

Table 14: **Continuation controls at** 800**M.** Shared 50 examples per task and length.

	S-NIAH-1				S-NIAH-2				S-NIAH-3			
Model	2k	4k	8k	16k	2k	4k	8k	16k	2k	4k	8k	16k
FS base	98	86	94	96	**100**	**98**	94	72	86	68	50	32
FS continuation	**100**	**100**	**100**	**100**	**100**	**98**	**98**	**84**	86	64	50	38
FS-L reported	**100**	90	86	84	98	**98**	94	80	**92**	**70**	**58**	**42**

Training from scratch. The reported CPT checkpoint has a higher RULER mean than the from-scratch model at all four lengths (Table 15). The gain comes from S-NIAH-2/3 and some multi-needle tasks, with a tradeoff on S-NIAH-1 and downstream accuracy. At 16k, the mean rises from 9.6 to 10.7, while S-NIAH-1 falls from 93 to 87. The eight-task downstream average falls from 52.2 to 51.0; CS in the table denotes this average.

At 800M, the tested from-scratch run reaches a downstream average of 56.8 but retrieves poorly: a 25-example S-NIAH-1 evaluation scores 32, 16, 8, and 0 at 2k, 8k, 16k, and 32k. CPT recovers long-range retrieval (Table 1). Good short-context performance therefore does not by itself establish a useful long-range branch. The experiments support FS initialization as a training recipe at both scales, but do not isolate why the from-scratch outcomes differ.

Table 15: **FS-L training recipes at** 340**M** (0–100). Both use pruning during evaluation, with 100 examples per RULER task and length. Bold and underline mark the better and second-best recipe.

	S-NIAH-1				S-NIAH-2				S-NIAH-3				RULER mean				CS
Recipe	2k	4k	8k	16k	2k	4k	8k	16k	2k	4k	8k	16k	2k	4k	8k	16k	Avg
From scratch	**100**	**97**	**98**	**93**	78	49	8	3	53	11	3	0	27.8	19.2	11.7	9.6	**52.2**
CPT (reported)	99	87	89	87	**99**	**72**	**39**	**18**	**70**	**16**	**8**	**1**	**32.3**	**21.4**	**15.5**	**10.7**	51.0

D.5 Retention capacity depends on future questions

FS-L must retain information before knowing which question will be asked. The adapted budget sweep measures the resulting accuracy–memory tradeoff. The multiple-question test checks whether the models can answer questions about several facts near the training length, a prerequisite for measuring retention capacity through this task.

The question distribution sets the budget. A small retained set can cover likely future questions when useful information is concentrated in a few cells. Consider instead F independent key–value facts in separate cells, with a later question choosing one uniformly. Keeping $C \leq F$ source cells gives direct access to only C/F of the possible targets. Contextual representations can copy information between cells, so this ratio is not an answer-accuracy bound. The accessible state includes all heads and layers, local buffers, and their contextual representations.

Even with compression, maintaining recovery accuracy above chance by a fixed amount on independent random values requires growing finite-precision memory, provided the question reveals only which value to retrieve. This motivates testing memory budgets and preserving more cells when information is dense. It does not identify the bottleneck in a trained model: a wrong answer can reflect lost information, failed routing, or failure to use available evidence.

Adapted retention budgets. We also continue FS with $P = 16$ and $P = 32$, keeping $W = 8$, $K = 4$, and the same 1500-step schedule. Each model evaluates with its training budget on the same 50 prompts (Table 16). At 2k, S-NIAH-1 falls from 100 at $P = 64$ to 98 at $P = 32$ and 82 at $P = 16$. Using the seed-42 controls, the broader curve is not monotone: at 16k the corresponding scores are 72–76, 64, and 72. S-NIAH-2/3 do not improve consistently with P. These independently adapted operating points measure a training–memory tradeoff, not the effect of changing the budget in one fixed model. Only $P = 64$ has repeated continuations. The local and temporary buffers also contribute to physical state, so changing P by a factor of four does not change total memory by that factor.

Table 16: **Adapted retention budgets at** 340**M.** Each model trains and evaluates with its stated P.

	S-NIAH-1				S-NIAH-2				S-NIAH-3			
Model	2k	4k	8k	16k	2k	4k	8k	16k	2k	4k	8k	16k
$P = 16$	82	68	74	72	**100**	**58**	22	**12**	56	14	4	**2**
$P = 32$	98	78	70	64	98	50	**28**	8	**74**	**20**	6	**2**
$P = 64$, run 1	**100**	**86**	**78**	**76**	94	46	18	2	62	12	**8**	**2**
$P = 64$, run 2	**100**	78	72	72	94	48	22	2	60	18	**8**	**2**

Figure 5 relates these scores to the streaming implementation's allocated attention buffers. The $P = 16/32/64$ configurations use $320.0/468.8/766.2$ MiB, unchanged across 4k, 64k, and 256k. Reducing P fourfold therefore gives a $2.4\times$ reduction in total buffers. Decode stays near 89 tokens/s; the smaller pool mainly saves storage (Table 17). Measurements follow the protocol in Appendix C, using the adapted $P = 16/32$ checkpoints and the reported $P = 64$ checkpoint. Buffer dimensions depend on architecture and budget. Accuracy uses native full-history evaluation; resources use bounded bf16 graph inference, whose numerical differences are checked separately.

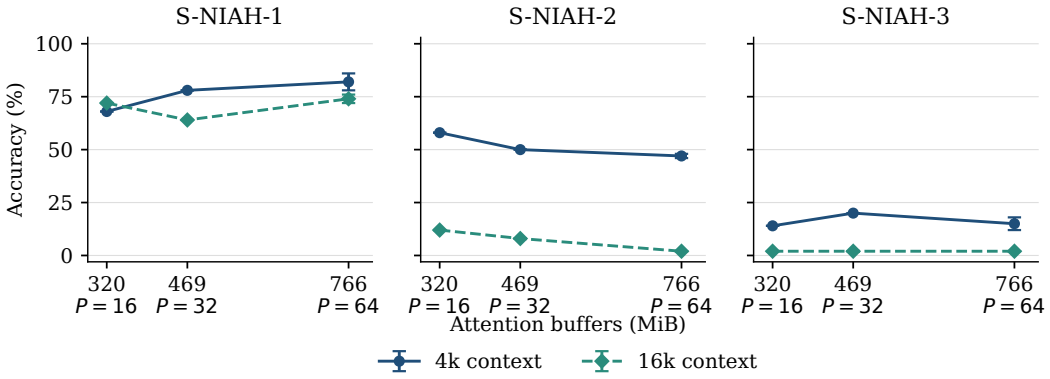

Figure 5: **Accuracy versus attention storage at** 340**M.** The $P = 64$ points show the mean and range of two seed-42 continuations.

Table 17: **Retention budget and bounded inference at** 340**M,** 256**k.**

P	Buffers (MiB)	Peak (GiB)	Prefill (Ktok/s)	Decode (tok/s)
16	**320.0**	**1.34**	2.13	88.4
32	468.8	1.49	**2.19**	**88.8**
64	766.2	1.77	2.17	88.3

Testing more facts at fixed context. We test FS and CPT FS-L on synthetic prefixes containing F distinct key–value facts. Each fact has a unique attribute, entity, and seven-digit value, with its source outside the final 528 prefix tokens. The question is revealed only after the prefix. Separate continuations start from independent copies of the same prefix state, so answers to earlier questions cannot help later ones. We use a RULER-style instruction and answer continuation, generate 16 tokens greedily, and score whether the target value appears. There are 50 shared prefixes per condition; we query every fact for $F \leq 8$ and eight uniformly sampled facts for $F = 16$. At 340M, prefixes contain 4253–4277 tokens, slightly above the training length. At 800M, they contain 3955–3971 tokens, with the prefix, question, and generated answer together fitting within 4096 tokens.

Table 18: **Multiple-question retrieval.** Accuracy (%) on shared prefixes.

Facts per prefix	1	2	4	8	16
Questions	50	100	200	400	400
340*M, 4253–4277 prefix tokens*					
FS	**94**	41	19	7.75	2.75
FS-L	90	**48**	**21**	**8.00**	**3.25**
800*M, strictly within 4096 tokens*					
FS	**100**	**51**	**27**	—	—
FS-L	96	**51**	23.5	—	—

Both scales pass a 90% single-fact competence threshold, but neither FS nor FS-L passes it with two or more facts (Table 18). The decline also occurs with unpruned access and, at 800M, entirely within the training context. It therefore does not isolate a pruning-capacity or length-extrapolation limit; query matching, binding, and fixed-K readout may also constrain performance.

D.6 Incremental retention gives bounded state

Query-independent priorities permit exact incremental maintenance of the retained set. This is what turns pruning into linear sequence work and bounded attention storage. Rescanning a growing prefix would not realize that complexity.

Let $\mathcal{P}$ be the full prefix and $\mathcal{D}$ the cells added at the next query-block boundary. Fixed priorities a_c and a common deterministic tie rule give

$$\mathrm{TopP}(\mathcal{P} \cup \mathcal{D}) = \mathrm{TopP}(\mathrm{TopP}(\mathcal{P}) \cup \mathcal{D}) \,, \tag{13}$$

A discarded cell already has at least P cells ahead of it; adding cells cannot improve its rank. Each cell enters once, so a size-P heap requires $O(M \log(P+1))$ total maintenance work over M cells and $O(P)$ retained-cell storage. A separate queue holds recent cells. This gives the incremental update used in Section 3.4.

Once a discarded cell leaves the recent queue, this head can never read it again. A streaming implementation can therefore keep at most $P + W$ completed cells, the local window, and unfinished-block buffers per head, assuming fixed parameters and budgets. Position counters still require $O(\log N)$ bits. This is the accessible state discussed in Appendix D.5. The streaming cache implements these updates with physical copies of retained cells, so evicted tensors do not keep full-prefix allocations alive. Appendix C reports allocated storage and throughput.

E FS-B: BALANCED TREE SEARCH

E.1 BALANCED TREE RETRIEVAL

We begin exploring retrieval in the high-information-density regime with FS-B (***FreeSolo-Balanced Tree Search***). Query-dependent tree search offers an intermediate $O(N \log N)$ routing cost between FS's $O(N^2/B)$ full-prefix scoring and FS-L's $O(N)$ pruning. We implement it with a streaming forest of balanced content trees.

Motivation. FS-L can work well in a low-information-density regime, where useful information is concentrated in a few cells and much of the prefix is redundant or irrelevant. Single-needle retrieval is one example. Query-independent pruning can preserve the useful cells while discarding much of the rest. This becomes harder when information is dense. Consider a list of unique key–value pairs with independent values, followed by a query asking for the value of a uniformly sampled key. The target is chosen only after the prefix has been processed. Every pair is equally likely to matter, so a small fixed pool cannot cover most queries as the list grows. Even with contextual compression, maintaining a fixed advantage over random guessing requires memory to grow with the independently queryable information (Appendix D.5). Code and mathematical text can pose a similar challenge when many definitions and dependencies may be needed later; these applications remain to be evaluated.

FS can access every eligible cell through query-dependent selection, but its $O(N^2/B)$ prefix scoring remains quadratic in sequence length. Our aim is a subquadratic sequence model that preserves this access in the high-information-density regime. FS-B therefore keeps a growing set of cells searchable until the query arrives. We still allow query-independent pruning, but use a norm threshold to remove likely noise rather than a top-P quota. When little can be discarded, most or all cells can enter the forest. Each query searches this memory with a fixed beam width and stopping rule (Figure 6). A branch skipped by one query remains available to later queries. The 512-token local RoPE branch and long-range NoPE readout remain as in FS. We describe the design below, then evaluate a fixed-search prototype without additional training in Appendix E.2.

Threshold admission and causal availability. For each layer ℓ and head h, FS-B admits

$$\mathcal{A}_i^\tau = \{c \in \mathcal{C}_i : a_c \geq \tau_{\ell h}\}, \tag{14}$$

using the maximum token-key norm a_c from Eq. (5). The threshold is either zero, admitting every eligible cell, or calibrated once from per-head norm quantiles on held-out batches before CPT and then fixed. It does not impose a quota on each prefix. Different heads can admit different cells, and a small norm alone does not certify that a cell is irrelevant to every future query.

Cells contain $s = 32$ tokens and query blocks contain $B = 128$ tokens. At each block boundary, newly completed cells become eligible for admission. The searchable forest and its summaries then stay fixed throughout the block, while each token searches independently. Training, prefill, and cached decoding use the same causal snapshots. Filtering a cell from long-range access does not remove its tokens from the local window.

Streaming tree construction. Each head maintains admitted cells in arrival order. When an aligned chunk of 2^k cells becomes complete, it builds a balanced content tree for that chunk once and caches it. Within a chunk, two power iterations estimate a principal direction, and median splits of the projected mean keys recursively partition the cells. Leaves are individual cells; neighboring leaves need not be adjacent in the sequence.

The binary decomposition of the current admitted count selects the active chunks: for example, $13 = 8+4+1$ gives three roots. A short chain of summary nodes joins these roots into the searchable tree. Thus grouping is based on content within chunks and arrival order across chunks. Completed chunk trees are reused across query blocks; only newly completed chunks and the chain joining the active roots need construction. For $M_i = |\mathcal{A}_i^\tau| > 0$, the resulting height is $H_i = \lceil \log_2 M_i \rceil$.

For the descendant cells $\mathcal{C}(v)$ of node v, the tree stores their mean key $\boldsymbol{\mu}_v$, count, an enclosing radius $\rho_v \geq \max_{c \in \mathcal{C}(v)} \left\| \bar{\mathbf{k}}_c - \boldsymbol{\mu}_v \right\|_2$, maximum norm $A_v = \max_{c \in \mathcal{C}(v)} a_c$, and earliest cell position $p_v^{\min} = \min_{c \in \mathcal{C}(v)} p_c$. Content-tree radii are computed directly; joining roots uses a count-weighted

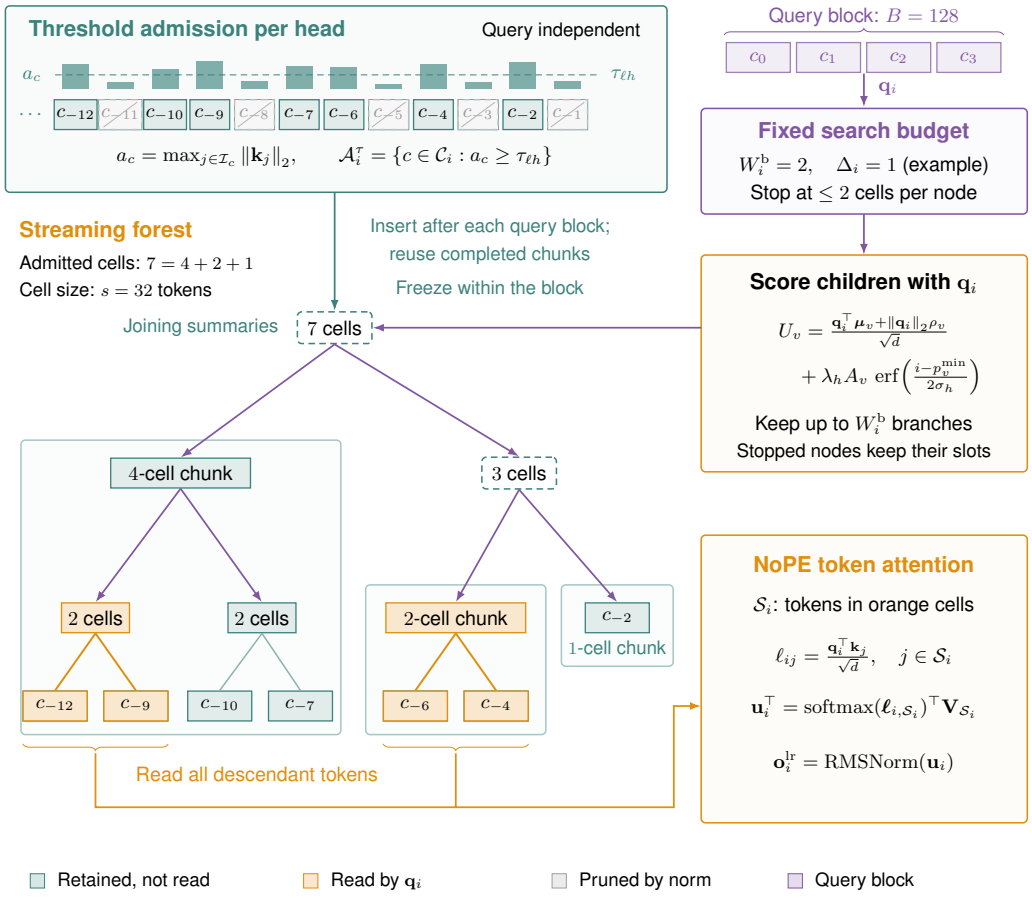

Figure 6: **Streaming FS-B selection.** Seven admitted cells form cached chunks of $4 + 2 + 1$ cells. Each head searches independently; orange cells supply the readout.

mean and expands the radius to enclose both children's balls. The default search priority is

$$U_v(\mathbf{q}_i) = \frac{\mathbf{q}_i^\top \boldsymbol{\mu}_v + \|\mathbf{q}_i\|_2 \rho_v}{\sqrt{d}} + \lambda_h A_v g_h(i - p_v^{\min}). \tag{15}$$

This bounds every descendant's FS cell score: the radius bounds its content term, while the maximum norm and earliest position bound the nonnegative, monotone erf term. Beam search remains approximate because discarding a branch with a smaller upper bound can still discard the best leaf. A centroid-scoring option omits the radius term and serves as a routing control.

Beam search and readout. Beam width W_i^{b} and stopping parameter Δ_i control the search budget. Here W_i^{b} is distinct from FS-L's recent-cell budget W. The preliminary experiments fix $W_i^{\mathrm{b}} = 4$ and $\Delta_i = 0$, so every surviving branch is followed to a leaf.

Starting at the root, search expands the current beam and keeps up to W_i^{b} nodes. A branch stops when its subtree contains at most 2^{Δ_i} cells; stopped nodes keep their beam slots, while the remaining slots take the highest-scoring children. Within a perfect chunk tree, this corresponds to stopping Δ_i levels above the leaves. The cell-count rule also applies to the unequal subtrees joining chunk roots.

Readout uses every raw token under the surviving nodes, without a further top-K cell selection. Thus the stopping rule controls the final softmax size, with at most $W_i^{\mathrm{b}} 2^{\Delta_i}$ cells per query. Their tokens enter the joint softmax and RMSNorm of Eq. (9); internal-node summaries are used only for routing. An empty tree gives zero long-range output. Deeper search reduces token reads but adds routing decisions, so neither speed nor accuracy is necessarily monotone in depth.

Learning the search budget. A learned controller and CPT with FS-B remain future work. A controller could choose $(W_i^{\mathrm{b}}, \Delta_i)$ from the query and tree statistics, spending more computation on queries that need broader access. Because beam membership and stopping are discrete, one option is a policy-gradient estimator (Williams, 1992) with a reward balancing next-token loss against node visits and token reads. CPT could then adapt the FS representations to this search rule. The experiments below instead use unchanged FS weights and a fixed search budget.

Computational cost. For a balanced tree, logical search and readout require $O(d[W_i^{\mathrm{b}} \log(M_i + 1) + sW_i^{\mathrm{b}} 2^{\Delta_i}])$ work per query. Stopping by subtree size keeps readout bounded as the tree grows. For fixed action menus, routing and token attention therefore take $O(N \log N)$ work, including the fixed local window. Streaming tree maintenance is separate: each admitted cell participates in $O(\log M)$ cached chunk trees. With $O(md \log m)$ construction for an m-cell chunk, these builds cost $O(Md \log^2 M)$ in total. The implementation also materializes index tables for traversal. Thus $O(N \log N)$ describes search and readout, rather than an established bound on this implementation's total runtime. The token archive uses $O(Nd)$ space; cached multiscale tree summaries use $O(Md \log M)$.

Relation to hierarchical memory and sparse attention. Hierarchical retrieval has several precedents. HCAM (Lampinen et al., 2021) and HSA (Hu et al., 2025a) score chunk summaries, attend within selected chunks, and combine their outputs. FS-B extends this coarse-to-fine access pattern with recursive search over a content tree, followed by a joint softmax over the retrieved tokens. HMT (He et al., 2025) recalls compressed segment memories within a recurrent architecture. FS-B instead preserves the token-level keys and values of admitted cells; its tree summaries guide access to those details.

Treeformer (Madaan et al., 2023) is a closer precedent for tree-based sparsity. It learns decision-tree partitions; its fine-grained variant attends to keys routed to the same leaf as the query, while its coarse variant combines value summaries along the query's path. FS-B constructs balanced partitions from the mean keys of causally available cells and searches several branches using query-dependent priorities. Streaming chunks index the growing prefix; beam width and the stopping rule determine how much of it each query reads. Together with threshold admission, this separates how much memory is kept from how much is searched and read. The aim is to retain access to dense information while adapting retrieval cost to the query. Whether this improves the quality–cost tradeoff remains to be tested.

E.2 PRELIMINARY EXPERIMENTS

We replace FS's full-prefix cell selection with tree search while keeping its weights unchanged. We set $\tau_{\ell h} = 0$ to admit every eligible cell, fix the beam width at $W_i^{\mathrm{b}} = 4$, and search to the leaves ($\Delta_i = 0$, $D_i = H_i$). Each query reads at most four cells, matching the FS and FS-L token-read budget. These experiments use neither CPT nor the learned search controller. All checkpoints were trained at 4k; evaluation at 8k and 16k therefore tests zero-shot extrapolation.

Table 19 reports official RULER S-NIAH accuracy on 100 examples per task and length, using the LLaMA tokenizer and greedy decoding. FS and FS-L scores are reproduced from Table 1; FS-L includes its reported CPT stage. These initial routing experiments use global content partitions. They test the search rule; streaming forest measurements must be evaluated separately because chunking changes the tree topology. The two entries identified in the caption are estimates, not evaluation results, and are excluded from rankings.

Table 19: **FS-B without additional training.** The FS-B S-NIAH-2 at 16k (341M) and S-NIAH-3 at 8k (743M) entries are estimates, excluded from ranking.

	S-NIAH-1				S-NIAH-2				S-NIAH-3			
Model	2k	4k	8k	16k	2k	4k	8k	16k	2k	4k	8k	16k
341M parameters												
FS	**99**	99	**100**	**100**	**100**	**77**	**41**	17	58	14	7	0
FS-L (CPT)	**99**	87	89	87	99	72	39	**18**	**70**	**16**	8	**1**
FS-B (no CPT)	**99**	**100**	**100**	**100**	95	70	34	16	51	**16**	**12**	**1**
743M parameters												
FS	97	91	**93**	**95**	**100**	**99**	**92**	72	85	71	51	34
FS-L (CPT)	**100**	93	83	84	99	**99**	89	**81**	**91**	**72**	**55**	**38**
FS-B (no CPT)	99	**96**	92	**95**	**100**	**99**	75	57	82	52	41	18

Without additional training, FS-B retains strong S-NIAH-1 retrieval through 16k: 99–100 at 341M and 92–99 at 743M. The harder tasks show the cost of approximate search. At 743M/16k, FS-B scores $57/18$ on S-NIAH-2/3, compared with FS's $72/34$ at the same read budget. Thus the existing FS representations support tree-based retrieval, although a four-branch beam does not preserve all of FS's recall. These results do not yet test retrieval among many independently queryable facts.

F AI USE AND REPRODUCIBILITY STATEMENTS

AI use statement. We used generative AI tools to assist with method implementation, theoretical analysis and proofs, experiment design, and interpretation of results. We also used them for figures, literature and reference checks, and manuscript editing. The authors take responsibility for the final manuscript and all AI-assisted text, claims, code, and figures.

Reproducibility statement. We will open-source the code and release checkpoint identifiers, data revisions, inference flags, and per-example outputs; Appendix A records the training and evaluation configuration.