

FreeSolo: Linear-time Sparse Attention with Zero-Shot Length Extrapolation

Zekun Shi^{1,3} Alastair James Arthur Price²

¹National University of Singapore

²University of Toronto ³Sea AI Lab, Singapore

zekun.shi@comp.nus.edu.sg

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 FREESOLO¹, 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. FS-B searches balanced trees with $O(N \log^2 N)$ algorithmic cost including radix-sort construction. FreeSolo demonstrates better zero-shot length extrapolation than the baselines without long-context fine-tuning.

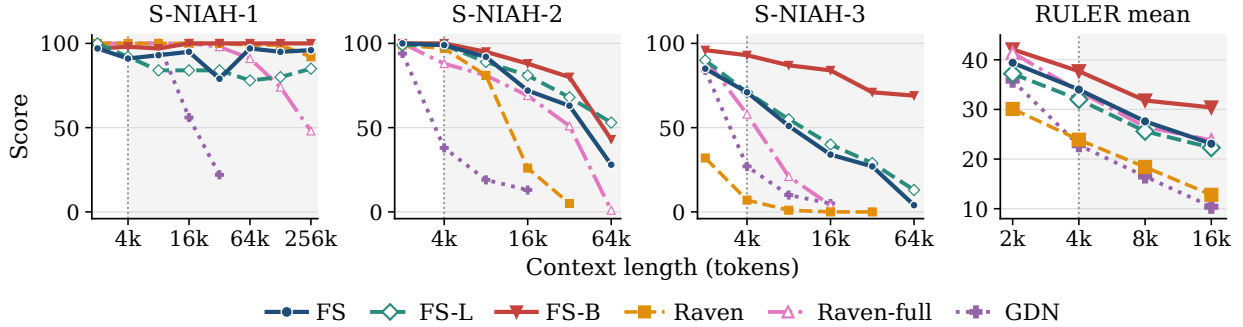


Figure 1: **Zero-shot RULER extrapolation at 800M.** Models train at 4k (dotted line); shading marks extrapolation. 100 examples per task and length.

1 Introduction

Long-context language modeling requires efficient attention and retrieval beyond the training length. Full attention has quadratic sequence cost (Vaswani et al., 2017). Recurrent models compress history into fixed-size states (Katharopoulos et al., 2020; Gu and Dao, 2023; Dao and Gu, 2024), while Native Sparse Attention

¹We will open-source the code.

(NSA) (Yuan et al., 2025) and Mixture of Block Attention (MoBA) (Lu et al., 2025) preserve token access but score growing sets of blocks. Neither compression nor sparsity alone ensures length extrapolation. Continued pretraining (CPT) on longer sequences can extend retrieval at additional training cost (Fu et al., 2024).

We introduce FREESOLO (FS), *a native sparse mechanism that learns token keys whose simple statistics make memory searchable, prunable, and hierarchically summarizable*. Mean keys give a simple chunk representation for routing and hierarchical search, while maximum token-key norms provide pruning priorities. These summaries need only identify evidence: top- K selection is followed by attention over the cells’ original token keys and values. FS scores all eligible cell summaries during training, but only selected token values enter the long-range readout. The read budget stays fixed as history grows. RoPE (Su et al., 2021) stays within a fixed local window, while long-range routing and readout use unrotated queries and keys, avoiding unseen rotary offsets (Figure 2).

We develop two main variants through short CPT from base FS at the same 4k context length. FreeSolo-Linear (FS-L) uses norm pruning for $O(N)$ attention cost and bounded state. FreeSolo-Balanced Tree Search (FS-B) keeps a growing archive and uses two content forests, with $O(N \log^2 N)$ algorithmic cost including radix-sort construction, superlinear but strictly subquadratic (Section 3.5). Both extrapolate zero-shot to 16k: at 800M, the RULER mean decreases only from 32.0 to 22.3 for FS-L and from 37.7 to 30.4 for FS-B relative to 4k. Notably, both retrieve single needles at 256k, $64\times$ the training length. At both scales, FS-B achieves the highest RULER mean in the superlinear group at 4k and beyond, and FS-L leads the linear group at every reported length (Table 1).

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 $\ell_{i,\mathcal{J}}$ and matching value rows $\mathbf{v}_j^\top$ in $\mathbf{V}_{\mathcal{J}}$. The readout is

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

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 $|\mathcal{J}_i| \leq m$, the readout costs $O(Nmd)$ per sequence, plus selection.

Length extrapolation. Zero-shot length extrapolation evaluates at $N > L$ after training at length L , with fixed weights. We measure retrieval rather than 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

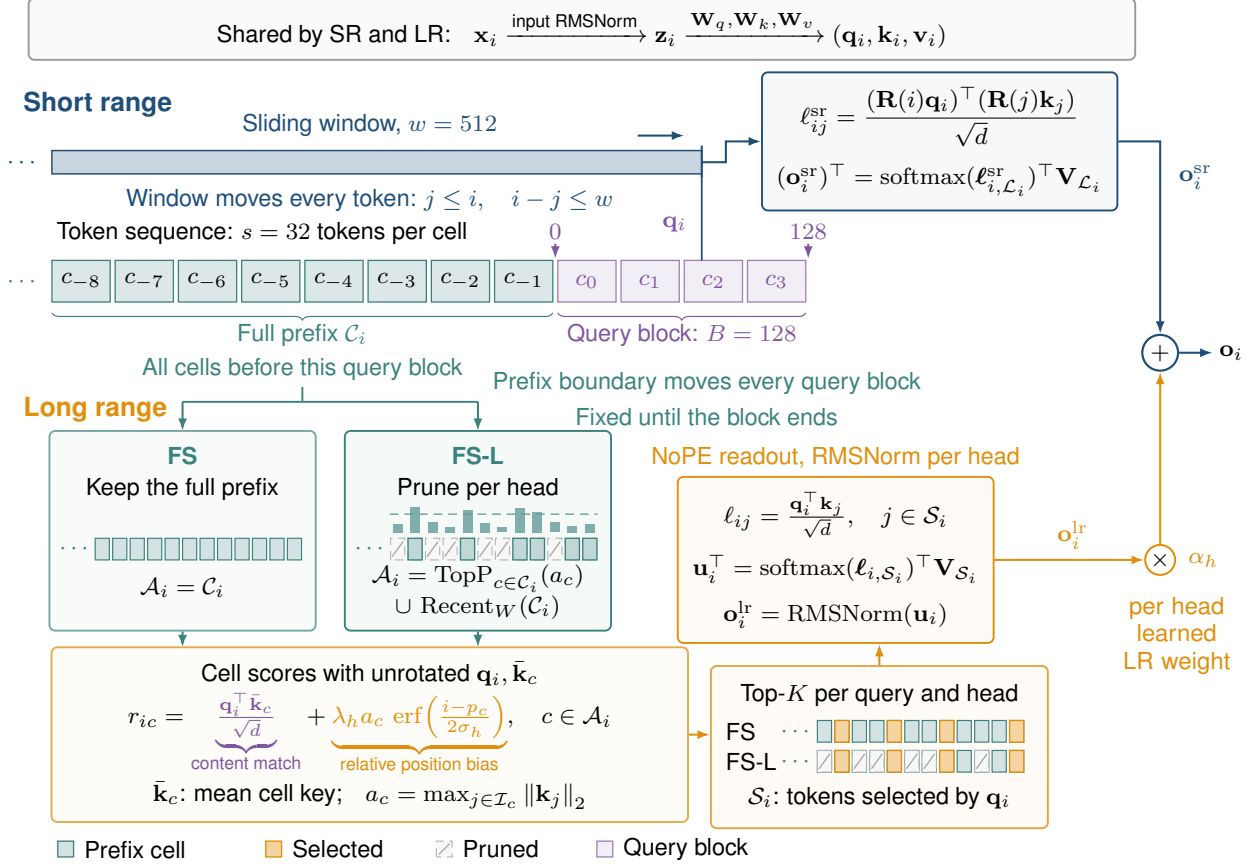


Figure 2: **FreeSolo attention**. FS scores the full prefix; FS-L prunes per head, independently of the query. Both select $K = 4$ cells per query and head. We use $P = 64$ high-norm cells, $W = 8$ recent cells, cell size $s = 32$, query blocks of $B = 128$ tokens, and SR radius $w = 512$.

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 and 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

$$\mathbf{o}_i = \mathbf{o}_i^{\text{sr}} + \alpha_h \mathbf{o}_i^{\text{lr}}, \quad (2)$$

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 $\mathcal{L}_i = \{j : 0 \leq j \leq i, i - j \leq w\}$, 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

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

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

$$(\mathbf{o}_i^{\text{sr}})^\top = \text{softmax}(\ell_{i,\mathcal{L}_i}^{\text{sr}})^\top \mathbf{V}_{\mathcal{L}_i}. \quad (4)$$

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.3).

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 $\mathcal{I}_c = \{cs, \dots, (c+1)s - 1\}$. Its summaries are

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

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:

$$\mathcal{C}_i = \{c \in \mathbb{Z}_{\geq 0} : (c+1)s \leq b_i\}. \quad (6)$$

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 $w = 512 \geq B - 1$; 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:

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

where $g_h(\Delta) = \text{erf}(\Delta/(2\sigma_h))$, $\sigma_h > 0$, 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 :

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

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

$$\mathbf{u}_i^\top = \text{softmax}(\ell_{i, \mathcal{S}_i})^\top \mathbf{V}_{\mathcal{S}_i}, \quad \mathbf{o}_i^{\text{lr}} = \text{RMSNorm}(\mathbf{u}_i). \quad (9)$$

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 and 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.

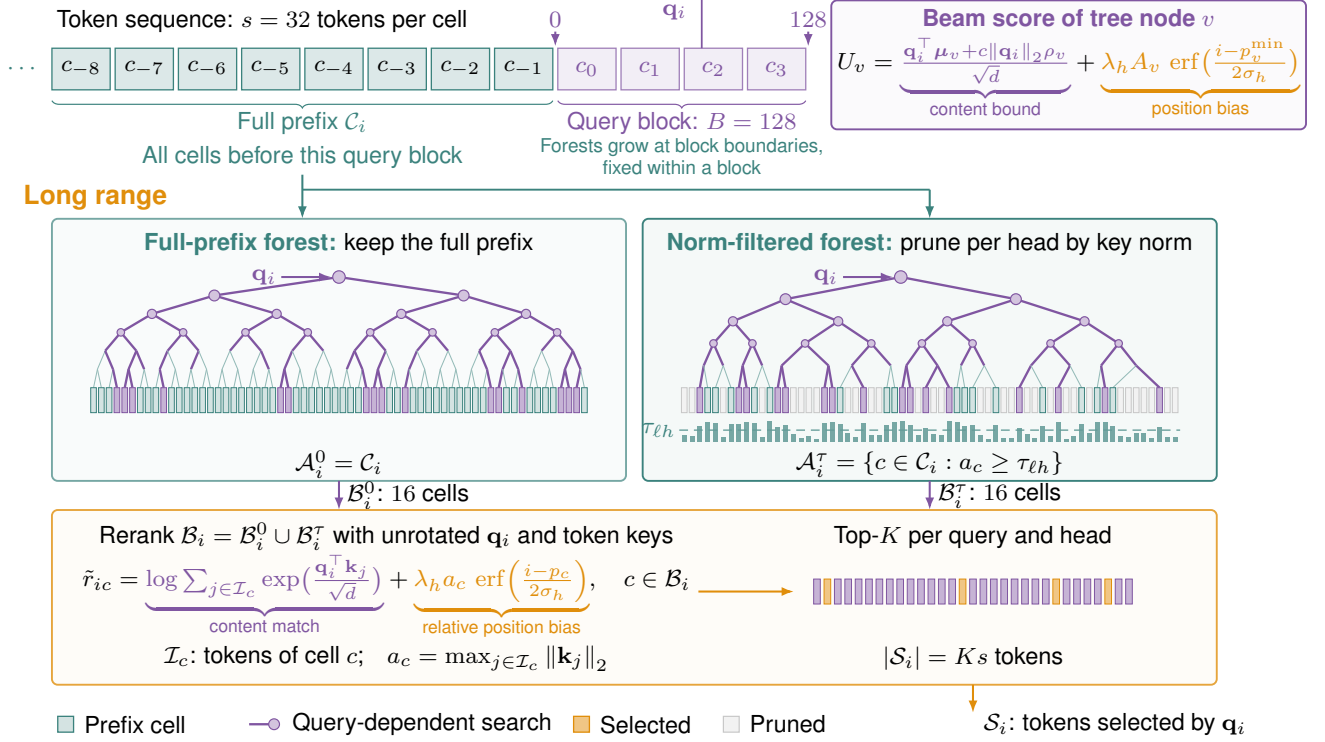


Figure 3: **FS-B long-range selection, per head.** The query searches two causal forests over the prefix: one over every cell and one over cells with $a_c \geq \tau_{\ell h}$. Each beam keeps 16 nodes per level down to the leaves, $\lceil \log_2 M \rceil$ levels for M admitted cells (6 and 5 in this 64-cell example). The union of both beams is reranked with token keys; the top $K = 4$ cells form $\mathcal{S}_i$, which is read as in FS.

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 Balanced tree search (FS-B)

FS-L suits evidence concentrated in a few cells, as in single-needle retrieval, because query-independent pruning can keep those cells. Dense, independently queryable facts are harder: in a list of unique key–value pairs followed by a question about a uniformly sampled key, every pair is equally likely to matter, so a fixed pool reaches a shrinking fraction of the possible targets (Appendix D.5). FS keeps every cell reachable but scores the full prefix in $O(N^2/s)$ work. FS-B keeps the local branch and the NoPE readout of Eq. (9) and changes only how $\mathcal{S}_i$ is chosen: each query searches content trees over a growing memory, so no cell is discarded for all future queries while search and read budgets stay bounded.

Two causal forests. FS-B always uses two forests per head, with admitted sets

$$\mathcal{A}_i^0 = \mathcal{C}_i, \quad \mathcal{A}_i^\tau = \{c \in \mathcal{C}_i : a_c \geq \tau_{\ell h}\}, \quad (10)$$

where $\tau_{\ell h}$ is the median cell norm on FineWeb-Edu windows, calibrated once per checkpoint and reused at all lengths. The first forest keeps the whole prefix searchable; the second emphasizes high-norm evidence, as FS-L does, without a fixed quota. In each forest, every aligned chunk of 2^k admitted cells is split recursively at the median of its mean keys along their principal direction, and summary nodes join the chunk roots, so M admitted cells give height $\lceil \log_2 M \rceil$ (Figure 9). A node v over cells $\mathcal{C}(v)$ stores their mean key μ_v , a radius $\rho_v \geq \max_{c \in \mathcal{C}(v)} \|\bar{\mathbf{k}}_c - \mu_v\|_2$, the maximum norm $A_v = \max_{c \in \mathcal{C}(v)} a_c$, and the earliest position $p_v^{\min} = \min_{c \in \mathcal{C}(v)} p_c$.

Beam search. From the root, each forest keeps the $W^b = 16$ highest-priority nodes per level down to the leaves, ranking nodes by

$$U_v = \frac{\mathbf{q}_i^\top \mu_v + c \|\mathbf{q}_i\|_2 \rho_v}{\sqrt{d}} + \lambda_h A_v g_h(i - p_v^{\min}). \quad (11)$$

For $c \geq 1$, U_v bounds the score of Eq. (7) for every cell below v : the radius bounds the content term, and A_v and $p_v^{\min}$ bound the nonnegative, monotone prior. The search remains approximate, since a branch with a lower bound can still hold the best cell; we use $c = 8$. The leaves reached form candidate sets $\mathcal{B}_i^0$ and $\mathcal{B}_i^\tau$ of at most 16 cells each, which differ because the forests group the prefix differently.

Token-level reranking. Mean keys can hide a single useful token, so FS-B rescores the deduplicated candidates $\mathcal{B}_i = \mathcal{B}_i^0 \cup \mathcal{B}_i^\tau$ with their token keys,

$$\tilde{r}_{ic} = \log \sum_{j \in \mathcal{I}_c} \exp\left(\frac{\mathbf{q}_i^\top \mathbf{k}_j}{\sqrt{d}}\right) + \lambda_h a_c g_h(i - p_c), \quad c \in \mathcal{B}_i, \quad (12)$$

and the top $K = 4$ cells form $\mathcal{S}_i$, read with Eq. (9).

Retrieval-weighted CPT. At 340M, FS-B reuses the FS-L weights. At 800M, base FS undergoes 0.42B-token CPT on 4096-token FineWeb-Edu sequences with a weighted next-token loss:

$$\mathcal{L} = -\frac{\sum_t w_t \log p_\theta(x_{t+1} \mid x_{\leq t})}{\sum_t w_t}, \quad w_t = 1 + \beta f_t + \beta_{\text{sw}} \chi_t, \quad (13)$$

Here $f_t, \chi_t \in \{0, 1\}$ are text-derived masks. The recall mask $f_t = 1$ marks starts and continuations of repeated spans, identified by a unique earlier four-token match more than 512 tokens back in the same document. The switch mask $\chi_t = 1$ adds a bonus at a recall start within 64 tokens of the preceding start if their earlier source positions are more than 64 tokens apart. Every token retains base weight 1. Training uses full-prefix FS selection with 16-cell reads; forests are used only at inference (Appendix E.4).

3.5 Computational cost

At fixed head dimensions and budgets, FS costs $O(N^2/s)$, FS-L $O(N)$, and FS-B $O(N \log^2 N)$ including radix-sort construction.

Cell statistics cost $O(Nd)$ per head. Each query scores $M \simeq N/s$ cells, giving FS a cost of $O(N^2 d/s + N(w + Ks)d)$ 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

$$O(Nd + M \log(P + 1) + N(C + w + Ks)d). \quad (14)$$

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 C).

FS-L results in Tables 1 and 4 use this bounded cache, which keeps only the retained cells (Appendix A); its resource measurements are in Appendix C. The efficiency comparison in Table 6 uses a masked implementation that stores $O(Nd)$ KV and rescans priorities at block boundaries, adding $\Theta(N^2/(Bs))$ work, with $O(N(P + W)sd)$ masked readout.

FS-B searches $M \leq \lfloor N/s \rfloor = \lfloor N/32 \rfloor$ admitted cells per forest. Beams descend $\lceil \log_2 M \rceil$ levels; search, reranking and readout cost $O(d[W^b \log M + W^b s + Ks])$ per query, or $O(N \log N)$ per sequence. Fixed-width radix sorting adds $O(Md \log^2 M)$ construction work, giving $O(N \log^2 N)$ combined algorithmic cost and $O(Nd)$ archive space per head. Appendix E.2 derives this bound and discloses repeated buffer copies in the reference streaming cache.

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. Training uses bf16 and AdamW (Loshchilov and Hutter, 2019); see Appendix A for architecture and optimizer settings.

FS-L uses 0.73B/1.57B CPT tokens at 340M/800M, about 5% of pretraining; FS-B uses 0.42B at 800M (Section 3.4). All CPT uses 4096-token contexts. Main-table baselines use pretraining checkpoints. Appendix D.4 adds selected 800M-scale baseline CPT controls, FS/FS-L comparisons at both scales, and seed and from-scratch studies.

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 algorithmic attention complexity at fixed head dimensions and budgets: $O(N^2/s)$ for FS, $O(N^2/b)$ for MoBA and NSA with a fixed block factor b , $O(N \log^2 N)$ for FS-B including radix-sort construction, and $O(N)$ for FS-L (Section 3.5), Raven, GDN, and SWA. NATS-L costs $O(NR)$ for R retained tokens, quadratic at a fixed retention fraction (Deng et al., 2026). Our InfLLM lookup is exhaustive.

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 81/40, compared with 26/0 for Raven and 72/34 for FS. FS-L reaches 53/13 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-B’s two-forest search (Section 3.4) improves retrieval further: at 800M and 16k it scores 100/88/84 on S-NIAH-1/2/3, with a full RULER mean of 30.4.

FS and FS-L also outperform Raven and NATS-L on the full RULER mean at both scales. At 800M/16k, the means are 23.1/22.3 for FS/FS-L, versus 12.8/2.8 for Raven/NATS-L. The gains extend beyond single needles: FS-L scores 23 on MK-NIAH-1 versus 3 for both linear baselines, and 17 on MV-NIAH versus 9/0.5 for Raven/NATS-L (Table 4). 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.

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.

4.3 Short-context quality

Table 2 reports eight zero-shot tasks with task definitions from the Language Model Evaluation Harness (Gao et al., 2023). 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. Both FS-L checkpoints retain similar short-context quality.

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

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. FS-B’s algorithmic cost includes radix-sort construction (Appendix E.2). Bold/underline mark the best/second-best scores within each block (super-linear or linear cost); dashes denote unavailable results.

Model	Params	Complexity	S-NIAH-1								S-NIAH-2						S-NIAH-3						Mean				
			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	–	–	–	–	<u>91</u>	24	0	0	–	–	38	1	0	0	–	–	26.2	13.3	0.2	0.0	
InfLLM	335M	$O(N^2)$	94	<u>58</u>	13	9	–	–	–	–	89	23	8	2	–	–	39	1	2	<u>1</u>	–	–	–	–	–	–	
MoBA	341M	$O(N^2/b)$	76	12	0	0	–	–	–	–	83	23	0	0	–	–	<u>59</u>	3	0	0	–	–	25.6	7.6	0.0	0.0	
NSA	513M	$O(N^2/b)$	43	23	0	0	–	–	–	–	48	18	0	0	–	–	1	0	0	0	–	–	13.2	7.3	0.9	0.0	
Raven-full	355M	$O(N^2)$	100	99	83	39	44	<u>4</u>	0	<u>0</u>	100	<u>89</u>	<u>41</u>	7	–	–	38	1	0	0	–	–	26.7	21.5	13.5	7.1	
FS	341M	$O(N^2/s)$	<u>99</u>	99	100	100	<u>98</u>	100	<u>98</u>	100	100	<u>77</u>	<u>41</u>	<u>17</u>	–	–	58	<u>14</u>	<u>7</u>	0	–	–	<u>30.8</u>	<u>23.2</u>	<u>16.5</u>	<u>11.8</u>	
FS-B	341M	$O(N \log^2 N)$	100	99	<u>99</u>	<u>99</u>	99	100	100	100	100	99	89	60	32	10	91	78	64	27	34	24	34.2	31.1	26.6	20.2	
FS-L	341M	$O(N)$	<u>99</u>	88	<u>90</u>	87	<u>65</u>	86	85	84	99	70	<u>34</u>	20	7	3	70	17	9	1	2	1	32.4	21.6	15.5	10.7	
Raven	357M	$O(N)$	95	96	93	<u>81</u>	78	<u>41</u>	<u>20</u>	–	<u>92</u>	70	36	<u>3</u>	–	–	3	0	0	<u>0</u>	–	–	20.5	<u>17.8</u>	<u>14.0</u>	<u>8.5</u>	
NatS-L	380M	$O(NR)$	100	<u>94</u>	37	19	0	0	0	–	89	<u>22</u>	20	<u>3</u>	–	–	<u>36</u>	<u>6</u>	1	<u>0</u>	–	–	<u>25.6</u>	15.3	7.4	4.0	
GDN	335M	$O(N)$	95	83	35	19	–	–	–	–	83	20	18	<u>3</u>	–	–	0	0	0	<u>0</u>	–	–	21.6	14.2	7.0	3.6	
SWA	341M	$O(N)$	18	15	7	3	–	–	–	–	31	21	6	2	–	–	25	2	<u>3</u>	<u>0</u>	–	–	14.6	9.3	4.5	4.0	
$\approx 800M$ scale																											
TF++	743M	$O(N^2)$	100	95	0	0	–	–	–	–	100	94	0	0	–	–	67	50	0	0	–	–	46.3	<u>37.5</u>	0.0	0.0	
InfLLM	743M	$O(N^2)$	100	96	20	4	–	–	–	–	100	89	12	4	–	–	77	50	4	0	–	–	–	–	–	–	
NSA	1128M	$O(N^2/b)$	100	75	8	0	–	–	–	–	<u>98</u>	46	3	0	–	–	30	7	0	0	–	–	40.7	24.8	2.6	0.1	
Raven-full	723M	$O(N^2)$	100	100	100	100	<u>98</u>	91	74	48	100	88	81	69	51	1	<u>91</u>	58	21	4	–	–	41.4	33.7	26.5	<u>23.8</u>	
FS	743M	$O(N^2/s)$	<u>97</u>	91	93	<u>95</u>	79	<u>97</u>	<u>95</u>	<u>96</u>	100	<u>99</u>	<u>92</u>	<u>72</u>	<u>63</u>	<u>28</u>	85	<u>71</u>	<u>51</u>	<u>34</u>	<u>27</u>	<u>4</u>	39.4	34.0	<u>27.6</u>	23.1	
FS-B	743M	$O(N \log^2 N)$	<u>97</u>	<u>98</u>	<u>97</u>	100	100	100	100	100	100	100	100	95	88	80	43	96	93	87	84	71	69	<u>42.1</u>	37.7	31.8	30.4
FS-L	743M	$O(N)$	100	<u>92</u>	<u>84</u>	<u>84</u>	<u>84</u>	<u>78</u>	<u>80</u>	<u>85</u>	<u>99</u>	99	89	81	68	53	90	71	55	40	29	13	37.2	32.0	25.6	22.3	
Raven	726M	$O(N)$	100	100	100	100	100	99	99	92	<u>99</u>	<u>97</u>	<u>81</u>	<u>26</u>	<u>5</u>	–	32	7	1	0	<u>0</u>	–	30.1	<u>23.9</u>	<u>18.4</u>	<u>12.8</u>	
NatS-L	793M	$O(NR)$	<u>86</u>	72	59	21	2	1	0	0	100	76	28	3	0	<u>0</u>	62	24	6	0	<u>0</u>	<u>0</u>	27.8	19.4	10.2	2.8	
GDN	694M	$O(N)$	100	100	100	56	22	–	–	–	94	38	19	13	–	–	<u>84</u>	<u>27</u>	<u>10</u>	<u>5</u>	–	–	<u>35.8</u>	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	

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.

These timings use bf16 parameters; Appendix C compares precision and execution paths.

The masked implementation 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; faster kernels for pruning, gathering, and sparse readout are future work (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 length generalization through an information-bottleneck view: FS routes through cell summaries, and FS-L also restricts memory before queries arrive (Appendix D).

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. The erf prior and local RoPE depend only on relative offsets, so shifting tokens, cells and query blocks together preserves the operator at fixed content vectors (Appendix D.1). Added tokens can still introduce competing cells.

Table 2: **Zero-shot downstream accuracy** (0–100). Rows are ordered by average within each scale; bold and underline mark the best and second-best scores.

Model	Params	HellaS.	PIQA	ARC-e	ARC-c	WinoG.	OBQA	SciQ	BoolQ	Avg
GDN	335M	<u>39.2</u>	<u>66.8</u>	55.8	28.2	<u>51.9</u>	<u>33.0</u>	83.2	59.8	52.2
FS	341M	38.9	<u>66.8</u>	57.6	25.9	50.9	32.4	81.2	61.1	<u>51.9</u>
TF++	335M	<u>39.2</u>	66.0	56.2	<u>28.1</u>	50.6	32.8	<u>81.5</u>	59.6	51.8
Raven	357M	37.6	65.2	55.8	27.6	51.2	32.2	80.6	<u>60.8</u>	51.4
SWA	341M	38.8	66.2	<u>56.4</u>	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
FS-B	341M	38.4	66.3	55.0	25.3	51.0	31.4	79.8	60.3	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	<u>51.9</u>	28.8	66.4	45.0	44.5
<i>≈ 800M scale</i>										
TF++	743M	48.9	70.9	63.9	<u>32.9</u>	54.3	34.4	87.4	61.9	56.8
FS	743M	<u>49.5</u>	<u>70.0</u>	<u>64.6</u>	31.7	56.0	34.4	86.7	58.8	<u>56.5</u>
Raven-full	723M	46.9	69.4	64.5	33.6	54.1	36.8	85.9	60.4	<u>56.5</u>
GDN	694M	47.4	69.1	64.8	32.8	53.8	<u>37.0</u>	<u>88.1</u>	58.2	56.4
SWA	743M	49.6	68.7	63.5	31.6	<u>55.1</u>	34.4	85.2	<u>61.8</u>	56.2
FS-B	743M	49.6	<u>70.0</u>	63.8	31.7	56.0	33.4	85.9	59.1	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 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 routing through these summaries encourages recognizable keypoints.

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 (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, but not dense, independently queryable facts (Section 3.4).

CPT adapts the model to this restriction. At 340M/4k, it recovers S-NIAH-1 from 62 under immediate pruning to 76–90 across runs (90 for the evaluated FS-L); unpruned continuations retain 100 (Appendix D.4). FS-L retains 14.3% of eligible cells at 340M/16k; Appendix D.3 separates retention, selection, and answering.

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). Since $g_h \geq \text{erf}(2) \simeq 0.9953$

for eligible cells, the term acts mainly as a norm bonus, so we do not credit extrapolation to its distance dependence (Appendix D.1).

6 Related Work

Selective recurrent memory. Mamba, Mamba-2, and GDN compress history in recurrent states (Gu and Dao, 2023; Dao and Gu, 2024; Yang et al., 2025b); GSA and Raven use softmax over memory slots (Zhang et al., 2024; Afzal et al., 2026). FS preserves token keys and values; summaries locate evidence. Mixture-of-Experts and Mixture-of-Memories route to parameter experts or recurrent states (Shazeer et al., 2017; Du et al., 2025); FS routes to token cells.

Native block-sparse attention. MoBA (Lu et al., 2025) also routes by mean keys and reads selected tokens, always including the causal current block. NSA (Yuan et al., 2025) learns compressed keys and values, derives selection scores from compressed attention, and includes compressed values in its readout. FS adds norm priorities and local-only RoPE to mean-key routing and selected-token readout. Training scores all eligible cells but reads only selected values. FS-L prunes before query matching; FS-B searches key hierarchies over the archive. Both reuse learned token keys across memory and search budgets without a learned chunk compressor.

Sparse token retrieval. Landmark Attention uses learned landmarks (Mohtashami and Jaggi, 2023); InFLM uses representative tokens (Xiao et al., 2024a). DRT (Hu et al., 2025b) and RAMba (Hu et al., 2025a) learn chunk retrieval and extrapolate beyond training length; routing scores weight their chunk outputs. FS hard-selects by fixed mean/norm summaries, then applies one softmax over selected tokens. Keys learn through token readout and local attention, without gradients through selection indices (Appendix E.5).

KV-cache selection and pruning. QUEST selects pages using key bounds (Tang et al., 2024). H₂O and SnapKV prune using observed attention (Zhang et al., 2023; Li et al., 2024); Devoto et al. (2024) retain low-norm keys. FS-L retains high-norm cells and adapts representations to that fixed rule through CPT. Retention needs neither a future query nor accumulated attention scores. NATS-L combines retained-token attention with GDN (Deng et al., 2026); its softmax history can grow. StreamingLLM keeps sinks and recent tokens (Xiao et al., 2024b); FS-L can retain cells from anywhere in the prefix.

Sparsity and length generalization. ASentmax studies sparse weights and extrapolation (Vasylenko et al., 2026); FS uses softmax over selected tokens. Position interpolation and YaRN rescale RoPE (Chen et al., 2023; Peng et al., 2024). RoPE/NoPE hybrids (Yang et al., 2025a; Puvvada et al., 2025) combine local RoPE and global NoPE layers; Kimi Linear and Kimi K3 combine recurrence with global NoPE and train on long contexts (Kimi Team, 2025; 2026). FS couples local RoPE with sparse NoPE retrieval within each head, training at short context throughout. Its long-range branch never encounters unseen rotary offsets.

7 Limitations

Experiments stop at the 800M scale. FS-L’s fixed retention budget can discard evidence before a query arrives. Its current CPT recovers S-NIAH-1 lost to immediate pruning, but the resulting 800M model has a lower

RULER mean than base FS at every evaluated length (Table 12). Finding a CPT objective that improves the suite mean within FS-L’s memory budget remains open. Retrieval-weighted CPT, as used for FS-B, motivates this direction (Appendix E.4). FS-B keeps growing memory and improves the RULER mean, but trails Raven-full on 800M multi-query and multi-value retrieval (Table 4). Learning its search policy, rather than using a fixed inference algorithm, is another direction for future work.

8 Conclusion

FreeSolo’s central thesis is that memory summaries need only locate evidence; the evidence can remain in original token values. Mean summaries of learned token keys support hierarchical search, maximum token-key norms support pruning, and local-only RoPE keeps distant retrieval free of unseen rotary offsets. FS-L and FS-B show how this representation supports bounded memory or a growing archive after short CPT, with zero-shot retrieval at 16k and single-needle recall at 256k from 4k training. This separates learning useful memory keys from choosing the capacity, search effort, and read bandwidth of the memory.

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)*, pages 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>.

- Justen Du, Weigao Sun, Disen Lan, Jiayi 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, pages 14125–14134, 2024. URL <https://proceedings.mlr.press/v235/fu24d.html>.
- Leo Gao et al. A framework for few-shot language model evaluation, 2023. URL <https://zenodo.org/records/10256836>.
- 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)*, pages 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 Krirman, 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*, pages 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*, pages 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. URL <https://arxiv.org/abs/2607.24653>.

- 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 Krizan, 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*, pages 47901–47911, 2024. URL <https://proceedings.mlr.press/v235/tang241.html>.

- 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>.
- 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. H₂O: 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.

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 base FS checkpoints are at 28,000 steps for 341M and 60,000 steps for 743M, corresponding to 14,680,064,000 and 31,457,280,000 training tokens. We refer to these scales by the rounded 15B/32B budgets. The smaller run makes about 2.2 passes over a 6.7B-token FineWeb-Edu subset; the larger run uses 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} . At 341M, the schedule warms up for 150 steps with a 1500-step cosine horizon, and we use the step-1400 checkpoint, selected on S-NIAH-1 (100 examples); the 1500-step continuations in Appendix D.4 show the run-to-run range. At 743M, it warms up for 300 steps and we use the final step-3000 checkpoint. These correspond 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. FS-L runs its bounded cache ($P = 64$, $W = 8$) with CUDA-graph decoding, FP32 parameters and BF16 autocast on A100 GPUs with 40 GB memory. MoBA and NSA recompute the full forward pass for each generated token. 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, evaluated with the LM Evaluation Harness.

Baselines. At 340M, NATS-L uses 21 layers and 64-token routing blocks. Table 3 lists the evaluated pretraining checkpoint of every model; all runs use 4096-token sequences. 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. Raven-full is the variant with full NoPE attention (Section 4.1). TF++ uses full RoPE attention, SWA a 512-token RoPE window, and InfLLM the TF++ checkpoint. NSA uses the paper’s configuration (compression blocks of 32 tokens with stride 16, 16 selected 64-token blocks, a 512-token window, and four key/value heads). Its larger parameter counts come from the key and value token-compression MLPs, which hold 208M parameters at 340M and 467M at 800M; NSA is 172M and 385M larger than the matching MoBA or FreeSolo backbone. For 800M GDN, 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.

Table 3: **Evaluated pretraining checkpoints.** Steps and training tokens at 524,288 tokens per step. MoBA is trained only at the smaller scale. FS-L and FS-B continue from the FS checkpoints: FS-L adds 0.73B (340M) and 1.57B (800M) CPT tokens; FS-B uses the FS-L weights at 340M and 0.42B retrieval-weighted CPT tokens at 800M (Appendix E.4).

Model	340M		800M	
	Steps	Tokens	Steps	Tokens
FS	28,000	14.68B	60,000	31.46B
TF++	28,610	15.00B	60,000	31.46B
SWA	28,000	14.68B	60,000	31.46B
MoBA	28,000	14.68B	–	–
NSA	28,000	14.68B	57,500	30.15B
GDN	28,610	15.00B	61,035	32.00B
NAtS-L	28,610	15.00B	61,035	32.00B
Raven	28,610	15.00B	61,035	32.00B
Raven-full	28,610	15.00B	61,035	32.00B

B Full RULER task breakdown

Table 4 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-B has the highest super-linear suite means from 4k on at both scales (Table 1) but remains below Raven-full on 800M MQ- and MV-NIAH.

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; ties share a mark. NAtS-L’s linear-time grouping assumes a bounded retained-token count (Section 4.1).

Table 4: **RULER results by task** (0–100) at 340M and 800M. Bold and underline mark the best and second-best scores within each group (super-linear or linear cost) and scale. Dashes denote unavailable results.

Task	Super-linear												Linear											
	Raven-full				FS				FS-B				FS-L				NaTS-L				Raven			
	2k	4k	8k	16k	2k	4k	8k	16k	2k	4k	8k	16k	2k	4k	8k	16k	2k	4k	8k	16k	2k	4k	8k	16k
<i>≈340M scale</i>																								
S-NIAH-1	100	99	83	39	<u>99</u>	99	100	100	100	99	<u>99</u>	<u>99</u>	<u>99</u>	88	<u>90</u>	87	100	<u>94</u>	37	19	95	96	93	<u>81</u>
S-NIAH-2	100	<u>89</u>	<u>41</u>	7	100	77	<u>41</u>	<u>17</u>	100	99	89	60	99	70	<u>34</u>	20	89	22	20	3	<u>92</u>	70	36	<u>3</u>
S-NIAH-3	38	1	0	<u>0</u>	<u>58</u>	<u>14</u>	7	<u>0</u>	91	78	64	27	70	17	9	1	<u>36</u>	<u>6</u>	1	<u>0</u>	3	0	0	<u>0</u>
MK-NIAH-1	<u>25</u>	<u>22</u>	<u>18</u>	<u>18</u>	27	20	15	5	27	25	19	24	31	24	21	10	<u>17</u>	<u>18</u>	<u>10</u>	<u>4</u>	11	11	<u>10</u>	3
MK-NIAH-2	0	1	0	<u>0</u>	<u>1</u>	<u>0</u>	0	<u>0</u>	2	<u>0</u>	0	1	0	1	0	0	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>
MK-NIAH-3	0	0	0	<u>0</u>	<u>1</u>	<u>0</u>	0	1	2	0	0	<u>0</u>	0	0	0	0	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>
MQ-NIAH	<u>20.8</u>	13.3	6.8	<u>5.0</u>	20.2	<u>20.8</u>	<u>12.8</u>	1.8	23.8	22.5	18.5	12.5	21.8	21.5	<u>10.5</u>	3.8	<u>17.5</u>	15.2	4.8	2.8	15.0	<u>16.5</u>	13.2	<u>3.0</u>
MV-NIAH	19.3	17.8	6.8	6.5	<u>23.2</u>	<u>18.5</u>	<u>15.5</u>	<u>9.8</u>	24.5	22.0	23.5	21.0	24.3	19.8	16.3	7.3	<u>18.8</u>	8.8	4.5	<u>7.0</u>	11.2	<u>12.0</u>	<u>12.0</u>	4.5
VT	0.8	1.4	0.4	0.2	<u>2.2</u>	<u>1.2</u>	<u>0.6</u>	1.6	3.2	<u>1.2</u>	1.0	<u>1.0</u>	<u>2.8</u>	0.0	<u>0.0</u>	<u>0.0</u>	4.8	<u>0.6</u>	<u>0.0</u>	<u>0.0</u>	2.6	2.0	1.4	1.0
CWE	0.2	0.6	0.0	0.1	<u>4.1</u>	<u>9.2</u>	<u>5.2</u>	<u>0.8</u>	5.2	10.3	6.5	1.8	4.9	9.1	4.0	0.4	<u>2.6</u>	<u>5.0</u>	<u>0.2</u>	<u>0.3</u>	1.0	0.2	<u>0.2</u>	0.1
FWE	0.0	0.0	0.0	0.0	<u>8.7</u>	<u>3.3</u>	<u>5.0</u>	<u>3.3</u>	9.0	6.0	11.7	5.0	9.7	<u>1.0</u>	2.3	2.7	<u>3.0</u>	1.7	<u>0.3</u>	<u>2.3</u>	0.0	0.0	0.0	0.0
QA-1	16	<u>13</u>	9	7	25	15	<u>5</u>	7	<u>23</u>	10	4	<u>4</u>	26	<u>8</u>	<u>4</u>	3	<u>19</u>	12	9	4	15	6	<u>4</u>	6
QA-2	–	–	10	10	–	–	8	5	–	–	<u>9</u>	<u>6</u>	–	–	<u>10</u>	4	–	<u>10</u>	10	–	–	–	12	<u>9</u>
Available-task mean	26.7	21.5	13.5	7.1	<u>30.8</u>	<u>23.2</u>	<u>16.5</u>	<u>11.8</u>	34.2	31.1	26.6	20.2	32.4	21.6	15.5	10.7	<u>25.6</u>	15.3	7.4	4.0	20.5	<u>17.8</u>	<u>14.0</u>	<u>8.5</u>
<i>≈800M scale</i>																								
S-NIAH-1	100	100	100	100	<u>97</u>	91	93	<u>95</u>	<u>97</u>	<u>98</u>	<u>97</u>	100	100	<u>92</u>	<u>84</u>	<u>84</u>	<u>86</u>	72	59	21	100	100	100	100
S-NIAH-2	100	88	81	69	100	<u>99</u>	<u>92</u>	<u>72</u>	100	100	95	88	99	99	89	81	100	76	28	3	<u>99</u>	<u>97</u>	<u>81</u>	<u>26</u>
S-NIAH-3	<u>91</u>	58	21	4	85	<u>71</u>	<u>51</u>	<u>34</u>	96	93	87	84	90	71	55	40	<u>62</u>	<u>24</u>	6	<u>0</u>	32	7	1	<u>0</u>
MK-NIAH-1	37	36	<u>29</u>	32	53	32	30	28	<u>50</u>	<u>34</u>	28	<u>31</u>	43	32	27	23	<u>31</u>	<u>20</u>	7	<u>3</u>	30	19	<u>14</u>	<u>3</u>
MK-NIAH-2	3	<u>1</u>	<u>0</u>	0	7	0	1	<u>1</u>	9	4	1	2	8	3	1	1	0	<u>0</u>	<u>0</u>	<u>3</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>
MK-NIAH-3	3	2	0	0	0	0	0	0	<u>2</u>	<u>1</u>	0	0	2	0	1	0	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>
MQ-NIAH	69.5	46.5	44.0	38.3	34.5	25.8	22.8	23.0	<u>45.8</u>	<u>28.3</u>	<u>28.8</u>	<u>27.3</u>	<u>26.2</u>	22.5	16.0	15.5	16.5	11.5	1.5	0.3	27.8	<u>19.8</u>	<u>11.0</u>	<u>7.2</u>
MV-NIAH	56.3	41.3	39.0	39.0	28.0	26.3	22.8	18.8	<u>38.3</u>	<u>30.3</u>	<u>29.8</u>	<u>26.3</u>	26.2	20.8	21.0	17.0	13.5	10.8	3.0	0.5	<u>25.8</u>	<u>17.0</u>	<u>13.5</u>	<u>9.0</u>
VT	0.6	<u>0.2</u>	<u>0.6</u>	<u>0.6</u>	<u>1.6</u>	<u>0.2</u>	0.4	<u>0.0</u>	3.8	1.4	1.0	0.6	3.8	0.8	<u>0.4</u>	<u>0.6</u>	0.0	<u>0.2</u>	0.0	0.0	<u>1.6</u>	0.8	0.6	1.0
CWE	3.3	9.3	5.4	1.2	8.4	26.7	13.5	<u>3.3</u>	<u>7.8</u>	<u>26.4</u>	<u>8.9</u>	<u>4.4</u>	4.5	13.0	7.1	1.2	0.4	0.7	0.0	0.0	<u>1.9</u>	<u>1.3</u>	<u>0.1</u>	<u>0.1</u>
FWE	1.7	1.7	0.0	0.0	<u>16.7</u>	<u>6.0</u>	<u>5.0</u>	<u>5.7</u>	17.7	8.7	7.0	8.0	0.3	0.0	<u>1.7</u>	<u>0.0</u>	<u>1.7</u>	<u>3.3</u>	4.7	<u>0.0</u>	18.0	11.0	1.3	3.7
QA-1	32	20	10	12	42	30	<u>12</u>	9	<u>38</u>	<u>27</u>	13	<u>10</u>	43	30	11	11	<u>23</u>	<u>14</u>	11	0	22	<u>14</u>	<u>6</u>	<u>5</u>
QA-2	–	–	<u>15</u>	<u>13</u>	–	–	<u>15</u>	11	–	–	17	14	–	–	19	16	–	–	<u>12</u>	9	–	–	11	<u>11</u>
Available-task mean	<u>41.4</u>	33.7	26.5	<u>23.8</u>	39.4	<u>34.0</u>	<u>27.6</u>	23.1	42.1	37.7	31.8	30.4	37.2	32.0	25.6	22.3	27.8	19.4	10.2	2.8	<u>30.1</u>	<u>23.9</u>	<u>18.4</u>	<u>12.8</u>

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 5 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. FS-B changes inference only: it uses the FS-L weights at 340M, and its CPT at 800M trains with FS selection, so it has no separate row. 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. Sparse long-range retrieval reduces scaling cost, but has overhead at the short training context.

Table 5: **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)	<u>228K</u>	<u>15.5</u>
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

Inference. Table 6 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. 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 stays nearly flat from 4k to 131k, where FS runs out of memory, while several baselines decode faster. FS-B runs to 131k within 19 GiB, but the measured reference uses unfused forest construction, traversal and reranking and repeatedly copies growing cache buffers (Appendix E.2). Its eager throughput (1.1–1.5K tokens/s prefill, 5.3–6.1 tokens/s decode) reflects this implementation rather than the $O(N \log N)$ search; with CUDA-graph decode it reaches 19.1–19.8 tokens/s. The benefit is sustained long-context retrieval with bounded query scoring, rather than the highest throughput at every length. Table 6 uses an FS-L implementation that stores all KV and rescans priorities (Section 3.5); Table 7 measures the streaming cache.

Table 6: **Prefill and eager decode throughput** at 340M scale on one A100-40GB (bf16, batch 1). OOM denotes the 40 GB memory limit.

Model	Prefill (Ktok/s)				Decode (tok/s)			
	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
FS-B	1.5	1.4	1.2	1.1	6.1	5.9	5.6	5.3

Compilation and CUDA graphs. With `torch.compile`, FS-L prefill reaches 62.8–68.2K tokens/s from 4k to 131k, and CUDA-graph capture raises its decode to 94.4–94.7 tokens/s through 64k; with the

same capture, SWA, GDN, and Raven decode at about 380, 275, and 262 tokens/s.

Bounded cache. The streaming implementation updates FS-L’s top- P pool exactly from newly eligible cells. Let $\mathcal{C}$ be the eligible prefix cells and $\mathcal{D}$ those newly eligible at the next query-block boundary. Fixed priorities a_c and a common deterministic tie rule give

$$\text{TopP}(\mathcal{C} \cup \mathcal{D}) = \text{TopP}(\text{TopP}(\mathcal{C}) \cup \mathcal{D}). \quad (15)$$

A discarded cell already has at least P cells ahead of it, so adding cells cannot improve its rank. Once it also leaves the recent queue and local window, its KV can be evicted.

The implementation keeps absolute cell indices for relative scores and computes RoPE locally; position counters require $O(\log N)$ bits. 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 7 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 7: **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)
<i>340M scale</i>				
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
<i>800M scale</i>				
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 6. Both implementations apply the same pruning rule, but precision and reduction order can change generated answers. On 25 prompts per S-NIAH task at 4k and 16k, bounded execution at 340M matches all 150 correctness decisions of the masked implementation (FP32 parameters, bf16 autocast). The bf16 CUDA-graph configuration used for timing matches 60/60 decisions at 340M and 59/60 at 800M on ten prompts per task and length, and 58/60 for the adapted $P = 16$ and $P = 32$ checkpoints; both scales retrieve all three tested 256k S-NIAH-1 examples. FS-L results in Tables 1 and 4 use the bounded cache on 100 examples per cell, with FP32 parameters and bf16 autocast (Appendix A). The FS-L continuation and adapted-budget accuracies in Appendices D.4 and D.5 also use bounded execution. The routing probes and eligible-prompt answer counts in Appendix D.3 use the full-history masked implementation, which applies the same pruning rule without evicting stored KV.

D Theory and ablation studies

In this section, we present additional theoretical analysis, ablation studies, and exploratory tests and visualizations (tagged *Probe*). The main findings are:

Ablations of short-range and long-range components (Appendix D.1).

1. (*Ablation*) Short-range RoPE is crucial for commonsense accuracy; single-needle length extrapolation persists without it.
2. (*Theory, Probe*) The long-range erf term acts mainly as a key-norm bonus; its residual distance dependence rarely changes selection.

Cell summaries and reading distance (Appendix D.2).

1. (*Theory, Probe*) Cell summaries need only locate useful evidence: the readout accesses the selected tokens’ original values directly, including needle reads over 166k tokens away in the traced prompt.
2. (*Theory*) Sparse readout limits interference from unrelated values; readout normalization alone cannot undo their mixing.

Norm pruning and retrieval coverage (Appendix D.3).

1. (*Probe*) Norm pruning substantially reduces query-scoring work: FS-L retains about one-seventh of eligible cells at 16k.
2. (*Probe*) On harder retrieval tasks, query-dependent selection remains a bottleneck even when the answer source survives pruning.

Continued pretraining (Appendix D.4).

1. (*Ablation*) Short CPT helps FS-L adapt to the pruning constraint, recovering much of the S-NIAH-1 accuracy lost when pruning is applied without adaptation.
2. (*Ablation*) Harder-task changes cannot be attributed to pruning alone: they vary across seeds and also occur with unpruned continuation.
3. (*Ablation*) Retrieval-weighted CPT improves the RULER mean of the full-prefix single-forest ablation (W32) at every evaluated length using only short-context training.

Retention budgets and multi-fact retrieval (Appendix D.5).

1. (*Theory*) Sustaining high accuracy when retrieving independently queryable random facts requires storage to grow with their number.
2. (*Ablation*) Adapted models do not improve monotonically with retention capacity; a larger pool is not sufficient for better retrieval.
3. (*Probe*) Multi-fact failures extend beyond pruning: unpruned and full-attention models also confuse values from the same prefix.

D.1 Ablations of short-range and long-range components

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 at $s = 32$ and $\sigma_h = 4$.

Position dependence. Local scores use $\mathbf{q}_i^\top \mathbf{R}(j - i) \mathbf{k}_j / \sqrt{d}$ with $0 \leq i - j \leq w$. 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 by a multiple of B 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 (short range without RoPE) replaces local RoPE with unrotated queries and keys and adds a $\log \operatorname{erfc}(\Delta / (2\sigma_h))$ bias with $\Delta = i - j$, retaining the 512-token window; it therefore changes both rotation and local bias. LR-content-only keeps local RoPE and removes the entire $\lambda_h a_c \operatorname{erf}((i - p_c) / (2\sigma_h))$ routing term, so long-range routing uses the query–mean-key similarity alone.

Table 8: **RULER ablations at 340M** (0–100). Bold and underline mark the best and second-best scores, including ties.

Model	S-NIAH-1							S-NIAH-2				S-NIAH-3			
	2k	4k	8k	16k	32k	64k	131k	2k	4k	8k	16k	2k	4k	8k	16k
FS	<u>99</u>	<u>99</u>	100	100	<u>98</u>	100	<u>98</u>	100	<u>77</u>	<u>41</u>	<u>17</u>	<u>58</u>	14	7	<u>0</u>
SR-NoPE	100	100	100	100	100	100	100	<u>98</u>	90	71	29	47	<u>23</u>	13	<u>0</u>
LR-content-only	95	97	<u>96</u>	<u>97</u>	95	<u>95</u>	96	80	57	38	14	71	25	<u>11</u>	1

Table 9: **Downstream accuracy of the ablations at 340M** (0–100), using the metrics in Table 2.

Model	HellaS.	PIQA	ARC-e	ARC-c	WinoG.	OBQA	SciQ	BoolQ	Avg
FS	<u>38.9</u>	66.8	57.6	<u>25.9</u>	<u>50.9</u>	32.4	<u>81.2</u>	61.1	<u>51.9</u>
SR-NoPE	28.4	55.7	36.9	23.7	49.2	25.6	48.2	38.7	38.3
LR-content-only	39.3	<u>66.1</u>	<u>57.1</u>	26.9	52.8	<u>31.8</u>	83.2	<u>60.0</u>	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-content-only retains 95–97 on S-NIAH-1 and a downstream average of 52.1, while worsening S-NIAH-2 and improving S-NIAH-3. The norm bonus is therefore not needed for single-needle extrapolation; it shifts S-NIAH-2 and S-NIAH-3 in opposite directions.

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

$r_{ic}^\infty = \kappa_{ic} + \lambda_h a_c$. Then

$$0 \leq r_{ic}^\infty - r_{ic} = \lambda_h a_c \operatorname{erfc}\left(\frac{i - p_c}{2\sigma_h}\right) \leq \lambda_h a_c \operatorname{erfc}\left(\frac{s}{4\sigma_h}\right). \quad (16)$$

At $s = 32$ and $\sigma_h = 4$, $\operatorname{erfc}(2) \simeq 0.004678$: 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 query i and a fixed available set with more than K cells, let γ_∞ be the gap between the K th 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 $\gamma_\infty > \delta_{\max}$, replacing the constant gate by g_h 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. Equivalently, routing can change only when $\gamma_\infty \leq \delta_{\max}$. It applies to FS’s full prefix and FS-L’s retained set at fixed representations.

Fixed-state routing controls. On the 16k native traces described in Appendix D.3, replacing g_h by one keeps 99.998% of the native top- K cells in the top- K set 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.

D.2 Cell summaries and 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 $\mathbf{k}_*$, its mean is $\mathbf{k}_*/s$, while $a_c = \|\mathbf{k}_*\|_2$. The content match is diluted by $1/s$, but the norm bonus preserves the cue’s magnitude. One key can thus make a cell visible to the router. A high-norm distractor receives the same bonus, so query alignment still decides which cells are read.

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

$$\mathbf{o}_i^{\text{lr}} = \frac{\mathbf{g} \odot \mathbf{y}}{\sqrt{\|\mathbf{y}\|_2^2/d + \epsilon Y^2}}. \quad (17)$$

When $\|\mathbf{y}/Y\|_2^2/d \gg \epsilon$ ($\epsilon = 10^{-8}$), the scalar softmax denominator cancels. Normalization therefore removes amplitude dilution, but irrelevant values can still alter the output direction. 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 and Cholak, 2022; Vasylenko et al., 2026).

Reading the needle. Figure 4 traces 340M FS and FS-L while answering the same 256k S-NIAH-1 prompt. Both generate the correct value, 2338687, from tokens 95,394–95,400 in a 262,010-token input. For a direct comparison, we display the same head in both models: layer 10, head 15, the head that maximizes the smaller

of the two models’ mean needle attention during the first answer. These are selected retrieval examples, not averages across heads or prompts.

The upper plots show the four cells read when predicting the first value token. A stem’s height is the cell’s total long-range softmax weight; orange marks the cell containing the value. The letters link each read to its decoded 32-token text, including unrelated cells. Each row gives the routing score r_{ic} from Eq. (7) for the models that selected the cell. These scores include the content term and erf prior; they are distinct from the attention weights plotted above. The lower plot follows attention to the exact source value as each answer token is predicted, excluding other tokens in its cell. In this example, the needle cell receives 89%/96% of the displayed head’s long-range softmax weight in FS/FS-L at the first value token. Both heads continue to read the distant value during generation. These weights precede readout RMSNorm and branch mixing; they do not establish a causal contribution to the answer.

Read distance across the prompt. Unlike Raven, which measures memory by the tokens written to recurrent slots (Afzal et al., 2026, Definition 4 and Figure 8), we measure where queries read. Selecting cell c at query i gives backward distance $i - p_c$ tokens. Figure 5 plots this distance against query position for the 800M checkpoints on the same 256k prompt. 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.

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. Both checkpoints read the needle cell in layer 1. At the final prompt query, 166,601 tokens from its center, layer 24 selects it in 6/16 FS heads and 2/16 FS-L heads. Over the final 2048 queries, averaging all layers and heads, 93.6% of FS selections and 93.1% of FS-L selections lie at least 131,072 tokens behind the query. These counts show direct access to distant cells, without measuring their attention weights.

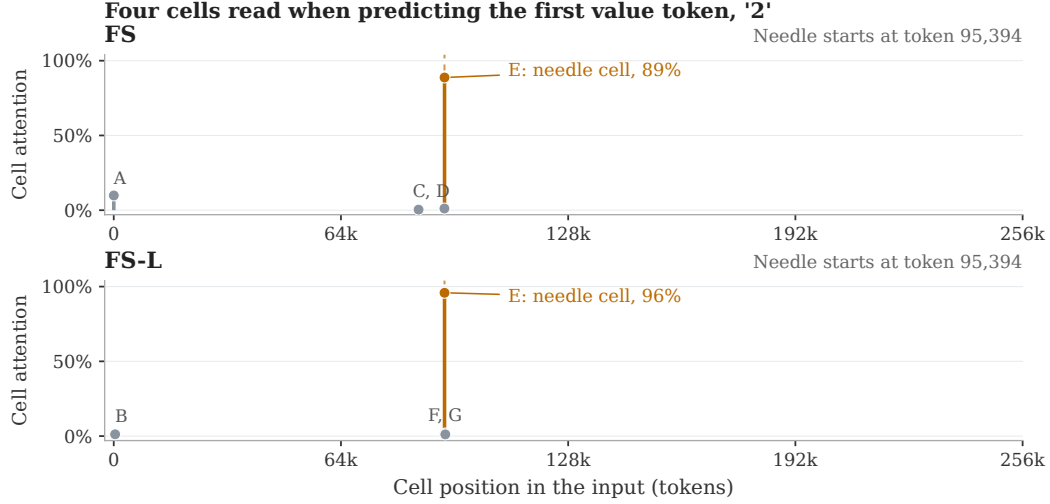
Retrieval on harder tasks. Figures 6 and 7 show selected successful S-NIAH-2/3 examples at 64k/32k with the same 800M checkpoints. Each pair uses the same prompt and needle position; both models generate the correct value. The common-head rule above selects layer 11, head 6 for both tasks. For S-NIAH-2, the needle cell near token 16,217 receives 72%/66% of this head’s long-range softmax weight in FS/FS-L at the first value token. For S-NIAH-3, the cell near token 7,591 receives 47%/83%. The displayed text shows the number or UUID prefix alongside the selected essay excerpts. The UUID spans two cells; only its first cell is read at the snapshot shown, and the next cell is selected later during generation. The lower plots follow attention to the full source value across the answer.

S-NIAH-1 | 256k context | 340M

Question at the end: What is the magic number for harmonious-uniform?

Correct value: 2338687

Both FS and FS-L generate this value correctly.



Text of the selected cells

Cell / first token Cell score r_{ic} Prompt text (32 tokens per cell)

A 0	FS: 13.07	<s> A special magic number is hidden within the following text. Make sure to memorize it. I will quiz you about the number afterwards. The grass
B 384	FS-L: 13.54	. The sky is blue. The sun is yellow. Here we go. There and back again. The grass is green. The sky is blue. The
C 87,904	FS: 12.78	The grass is green. The sky is blue. The sun is yellow. Here we go. There and back again. The grass is green. The
D 95,360	FS: 13.49	The sky is blue. The sun is yellow. Here we go. There and back again. One of the special magic numbers for harmonious-uniform is
E 95,392	FS: 15.43 FS-L: 16.40	: 2338687 . The grass is green. The sky is blue. The sun is yellow. Here we go. There and
F 95,552	FS-L: 13.61	The grass is green. The sky is blue. The sun is yellow. Here we go. There and back again. The grass is green. The
G 95,616	FS-L: 13.55	yellow. Here we go. There and back again. The grass is green. The sky is blue. The sun is yellow. Here we go. There

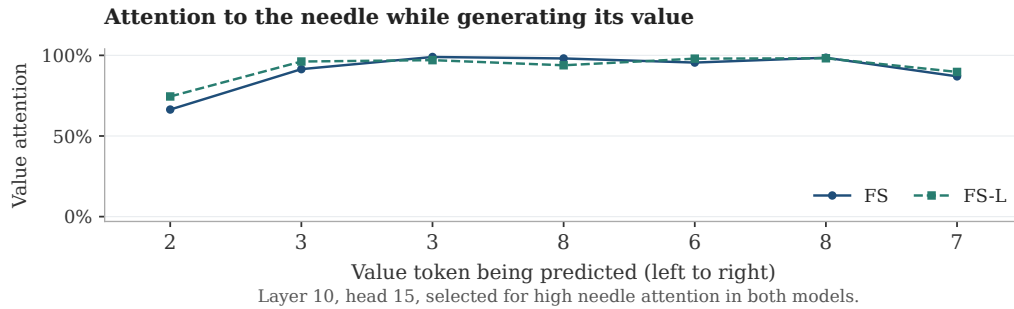


Figure 4: **Retrieving a number at 256k (340M).** Selected cells, routing scores, prompt text, and attention during generation. Both answers are correct.

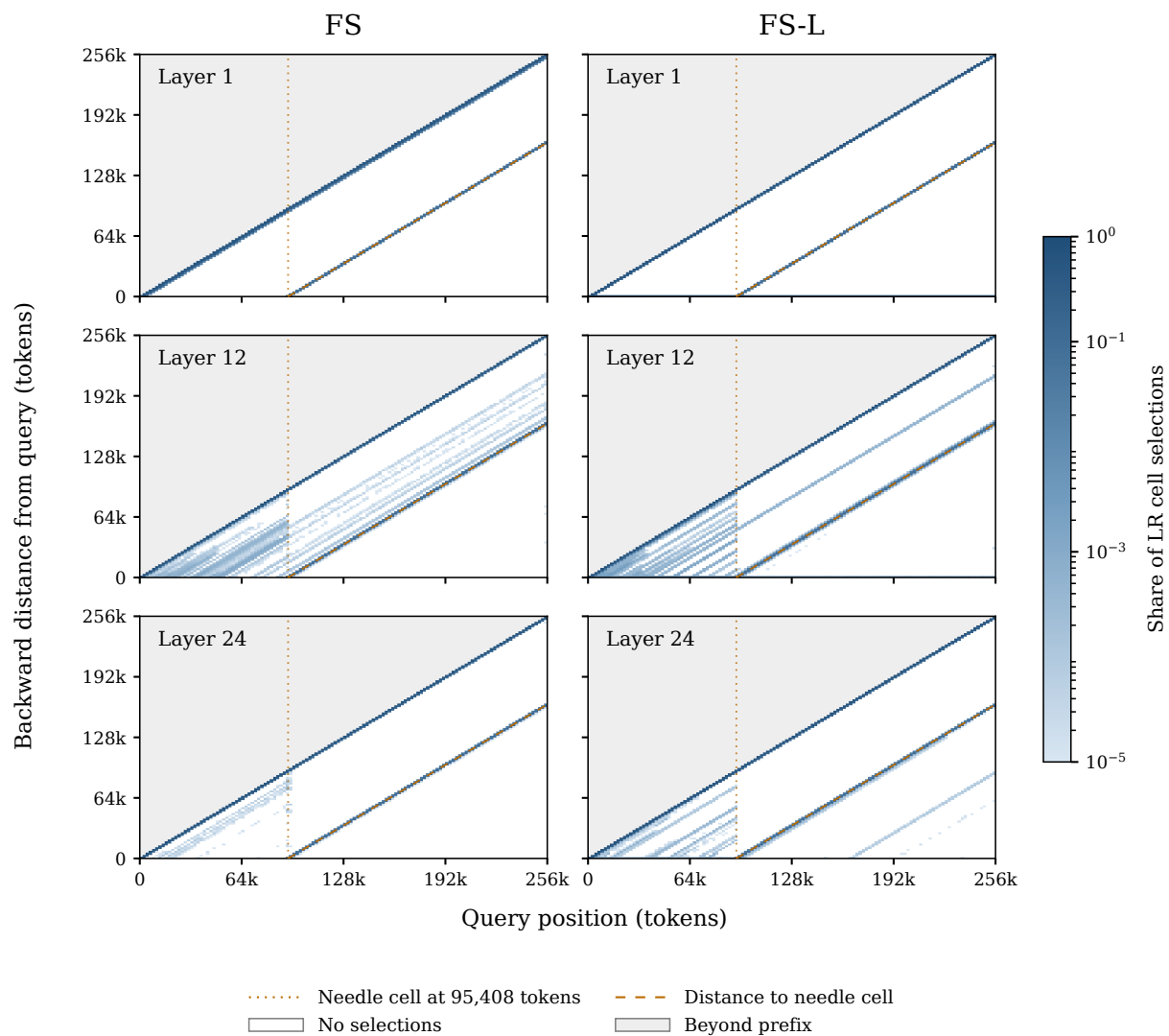


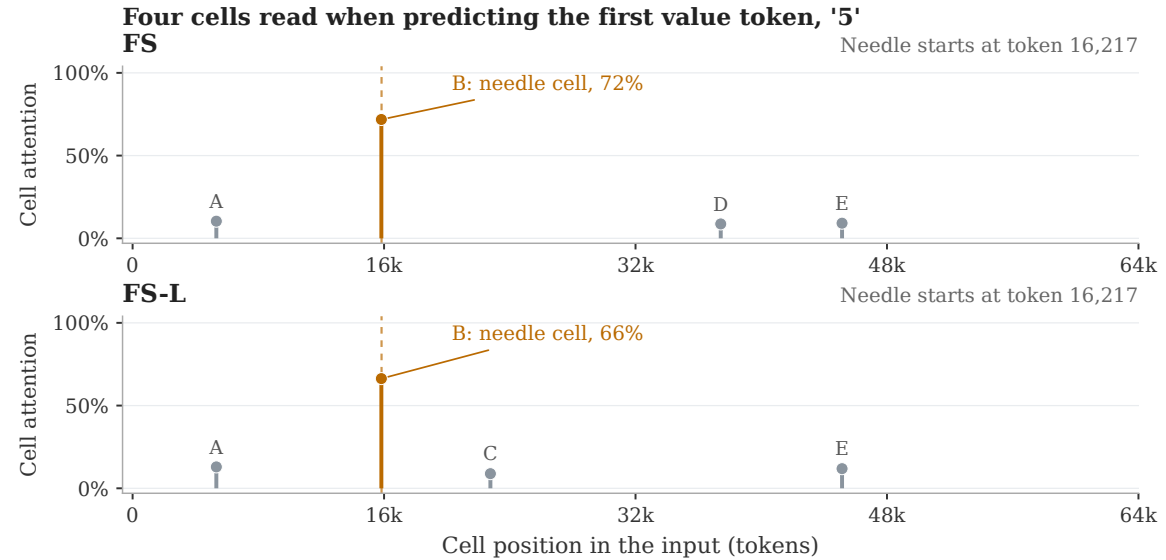
Figure 5: **Long-range read distance at 800M** on one 256k S-NIAH-1 prompt. Orange guides mark the needle position and its distance from each query.

S-NIAH-2 | 64k context | 800M

Question at the end: What is the magic number for roasted-poetry?

Correct value: 5107245

Both FS and FS-L generate this value correctly.



Text of the selected cells

Cell / first token	Cell score r_{ic}	Prompt text (32 tokens per cell)
A 5,440	FS: 20.50 FS-L: 20.23	. Apple itself did. Apple made microcomputers because that's what Steve Wozniak wanted for himself. He couldn't have afforded a
B 16,192	FS: 21.09 FS-L: 22.05	startup, you probably shouldn't do it. One of the special magic numbers for roasted-poetry is: 5107245
C 23,296	FS-L: 19.94	yourself what's good. You can't trust authorities. They'll lie to you on this one. Comment on this essay.January 2
D 38,304	FS: 21.38	8,000. A doctor, P. Decimus Eros Merula, paid 50,000 sestertii
E 46,208	FS: 21.49 FS-L: 20.19	One place this happens is in startups. At our startup we had Robert Morris working as a system administrator. That's like having the Rolling St

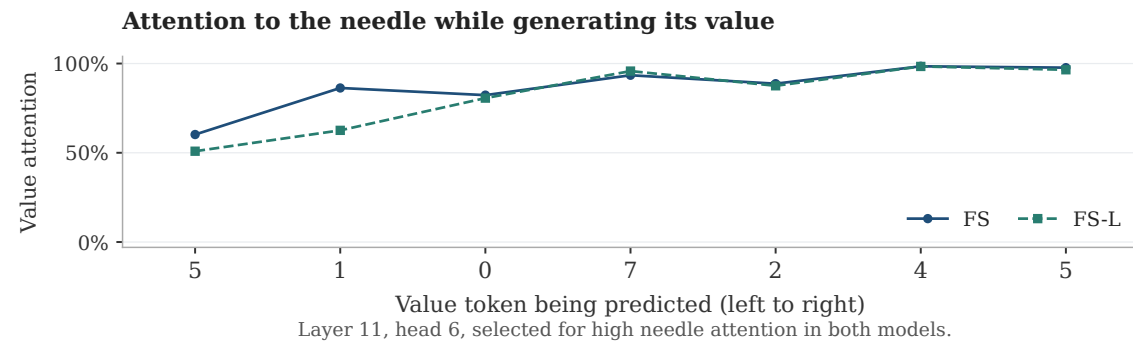


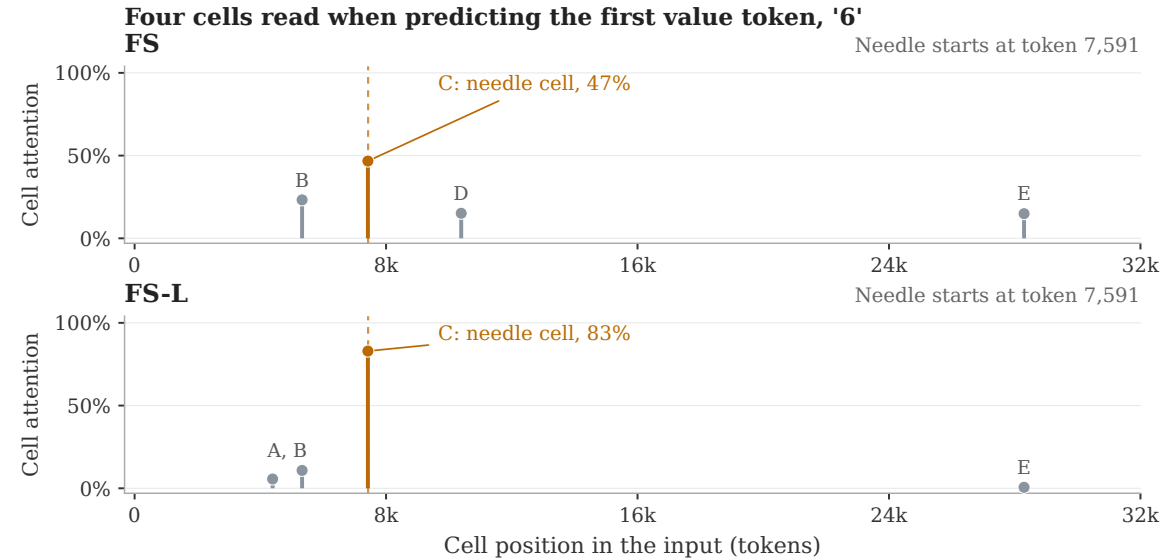
Figure 6: **Retrieving a number at 64k (800M).** Selected cells, routing scores, prompt text, and attention to the S-NIAH-2 value. Both answers are correct.

S-NIAH-3 | 32k context | 800M

Question at the end: What is the magic UUID for tested-shout?

Correct value: 6f4cc69a-4b22-4308-9c8e-aee95715bd6f

Both FS and FS-L generate this value correctly.



Text of the selected cells

Cell / first token	Cell score r_{ic}	Prompt text (32 tokens per cell)
A 4,480	FS-L: 18.59	in the App Store", and it's combined with the emotion "Really, it's Apple's fault." Another wrote: I believe that they
B 5,440	FS: 19.97 FS-L: 19.12	. Apple itself did. Apple made microcomputers because that's what Steve Wozniak wanted for himself. He couldn't have afforded a
C 7,584	FS: 20.13 FS-L: 18.97	tested-shout is: 6f4cc69a-4b22-4308-9c8e-aee
D 10,624	FS: 18.69	to know to understand what it meant. In Ansi Common Lisp I tried to move things along as fast as I could, and even so I didn'
E 28,960	FS: 19.50 FS-L: 18.58	= x (if foo 1 2))7. A symbol type. Symbols differ from strings in that you can test equality by comparing a pointer.

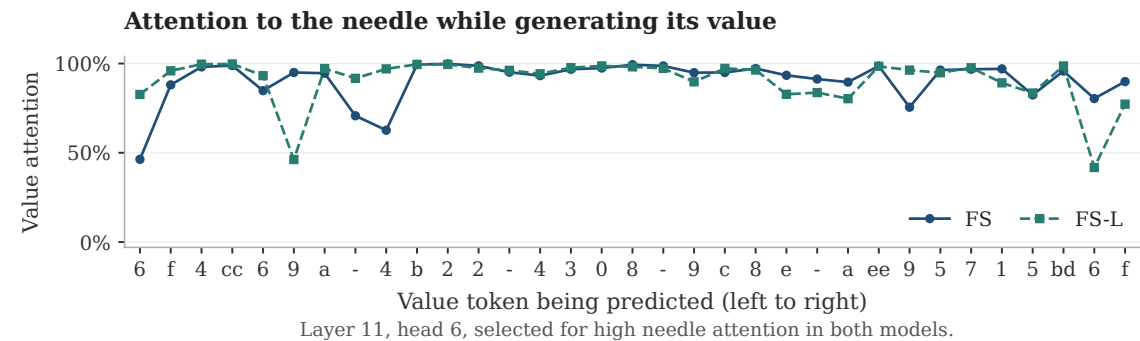


Figure 7: **Retrieving a UUID at 32k (800M).** Selected cells, routing scores, prompt text, and attention to the S-NIAH-3 value. Both answers are correct.

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.

Probe protocol. We trace the FS and FS-L checkpoints of Table 1 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. 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 10 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. These probes and the eligible-prompt answer counts below use the full-history masked implementation of FS-L, rather than the bounded cache used for the main accuracy tables.

Table 10: **Source-cell availability and selection at 16k.** The source is the cell containing the answer value. Rates average over all heads and the final 16 prompt positions within each prompt, then over prompts; FS (offline) applies the FS-L retention rule to FS keys.

Task	Prompts	Source retained (%)		Source selected (%)	
		FS (offline)	FS-L	FS	FS-L
340M <i>scale</i>					
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 <i>scale</i>					
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, FS-L improves source availability by 5.66/1.54/1.11 percentage points on S-NIAH-1/2/3 (computed before rounding). 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 checkpoints and do not separate 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. Source access does not

guarantee a correct answer: on the same eligible prompts, greedy answers are correct on 47/47, 9/50, and 0/47 for FS and 40/47, 9/50, and 0/47 for FS-L on S-NIAH-1/2/3 at 16k.

At 800M, the corresponding source-availability changes are $-1.04/0.82/0.13$ percentage 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. On S-NIAH-2/3, FS selects the source in only 1.3–3.6% of head/query pairs and FS-L selects a retained source in 6.6–13.4%, so selection rather than retention limits the harder tasks.

Compression. For each head, define the retained fraction $\eta_i = |\mathcal{A}_i|/|\mathcal{C}_i|$. 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 $P = 64$ and $W = 8$. These fractions concern cell counts; Appendix C measures physical storage.

D.4 Continued pretraining

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. These results support adapting an FS checkpoint to the pruning constraint through short CPT.

Learning with hard indices. As in Section 3.3, top- P and top- K indices receive no loss gradient, and we use no auxiliary retention loss or straight-through estimator. A key receives no gradient through an access that excludes it, and the remaining gradients need not increase its norm; scores and norm ranks change only through the shared projections and contextual states.

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 $P + W = 72$. Optimization settings are in Appendix A.

Continuation controls. Table 11 compares pruning and continuation at 340M using the same first 50 examples per task at 4k. Immediate pruning applies FS-L’s $P = 64$ retention rule without adaptation. The seeded controls start from the same FS checkpoint with a fresh optimizer, the same corpus and schedule, and 1500 steps (0.786B tokens); FS uses full-prefix routing and FS-L keeps pruning in training and evaluation. The main FS-L checkpoint uses 1400 steps. We repeat seed 42 with the same data order and run seed 43 once; changing the seed also changes data sampling.

At 4k, immediate pruning reduces S-NIAH-1 from 100 to 62. FS-L recovers 90, while the seed-42 adapted runs recover 20 and 14 percentage points (paired-prompt bootstrap 95% intervals $[4, 36]$ and $[0, 28]$). All three unpruned continuations retain 100. S-NIAH-2 declines under seed 42 even without pruning, whereas seed 43 reaches 82 with pruning versus 74 without it. Thus CPT supports S-NIAH-1 recovery, but harder-task changes vary across seeds and also occur without pruning. Intervals use 5000 prompt resamples and do not estimate seed uncertainty.

Harder-task gains also occur without pruning at 800M: after 3000 CPT steps (1.57B tokens), FS’s 16k S-NIAH-2/3 accuracy rises from 72/32 to 84/38 on 50 shared prompts per task.

Table 11: **Pruning recovery and seed variation at 340M.** Accuracy (%) on 50 shared prompts per task at 4k, the training context. FS-L is the 1400-step checkpoint of Table 1; seeded rows use 1500 CPT steps without (FS) or with (FS-L) pruning.

Model	S-NIAH-1	S-NIAH-2	S-NIAH-3
FS base	100	74	14
Immediate pruning	62	72	14
FS-L	90	64	14
FS, seed 42 (run 1)	100	50	16
FS, seed 42 (run 2)	100	52	20
FS, seed 43	100	74	6
FS-L, seed 42 (run 1)	82	44	12
FS-L, seed 42 (run 2)	76	46	18
FS-L, seed 43	82	82	8

Budget-matched continuation of unchanged architectures. To assess the effect of additional training alone, Table 12 reports selected 800M-scale baselines and FS continued from their Table 1 checkpoints with plain next-token loss. These controls match FS-L’s 3000 updates of 524,288 tokens each (1.57B total), using FineWeb-Edu at 4096-token context (Appendix A). The table also includes a separate single-forest width-32 ablation of FS-B’s recall-weighted recipe: 1600 updates of 262,144 tokens each (0.42B total, 27% of the matched budget), at the same context length (Appendix E.4). Each before/after pair uses identical prompts and task sets: means cover 12 tasks at 2k/4k and 13 at 8k/16k, following the protocol of Table 1.

Plain CPT does not improve the baselines or unpruned FS consistently across all evaluated lengths. Only FS-B’s recall-weighted CPT improves the RULER mean at all four lengths in Table 12, as measured by the single-forest width-32 ablation: 40.9/36.0/30.4/28.2 becomes 41.3/37.2/31.6/30.4. FS-L recovers much of the S-NIAH-1 accuracy lost under immediate pruning (Table 11), but its suite mean remains below base FS at every length in Table 12.

Table 12: **Only FS-B’s weighted CPT improves the RULER mean at all four lengths.** Suite-mean changes at 800M scale (+ CPT minus base, points, on the same prompts and protocols as Table 1; the base means are the Table 1 rows). Each “+CPT” arm continues the Table 1 checkpoint of the same architecture with plain next-token loss at the FS-L budget (1.57B tokens). The FS-L arm instead keeps the pruning loop in training at the same budget. The single-forest ablation uses FS-B’s recall-weighted CPT, spending 0.42B tokens (27%; Appendix E.4), and compares the full-prefix width-32 read before and after CPT (base means 40.9/36.0/30.4/28.2). FS-B itself denotes the two-forest design of Table 1.

Continuation arm	Δ 2k	Δ 4k	Δ 8k	Δ 16k
SWA, 743M, plain	−1.4	+0.4	+0.2	−0.2
Raven-full, 723M, plain	−7.2	−5.6	−1.4	+0.8
Raven, 726M, plain	−2.2	−1.0	−2.8	−1.9
NSA, 1128M, plain	−1.2	−2.2	+0.3	+0.1
FS, 743M, plain	−1.6	−1.3	−1.0	0.0
FS, 743M → FS-L (pruning)	−2.2	−2.0	−2.0	−0.8
Single-forest W32 ablation, 743M, weighted	+0.4	+1.2	+1.2	+2.2

Training from scratch. CPT gives a higher RULER mean than training from scratch at all four lengths (Table 13). The gain comes from S-NIAH-2/3, 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.

At 800M, training with pruning from scratch reaches a downstream average of 56.8 but S-NIAH-1 scores of only 32, 16, 8, and 0 at 2k, 8k, 16k, and 32k (25 examples), whereas CPT retrieves well (Table 1); we therefore use CPT at both scales.

Table 13: **FS-L training recipes at 340M (0–100)**. Both use pruning during evaluation, with 100 examples per RULER task and length. Bold marks the better recipe.

Recipe	S-NIAH-1				S-NIAH-2				S-NIAH-3				RULER mean				Down.
	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	99	88	90	87	99	70	34	20	70	17	9	1	32.4	21.6	15.5	10.7	51.0

D.5 Retention budgets and multi-fact retrieval

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.

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 $m \leq F$ source cells gives direct access to only m/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.

Compression does not remove this limit. If the F values are independent and uniform over V outcomes, the question reveals only which value to retrieve, and the state holds b bits, Fano’s inequality bounds the mean accuracy by $\bar{p} \leq (1 + b/F)/\log_2 V$; any fixed accuracy with $\bar{p} \log_2 V > 1$ therefore needs $b = \Omega(F)$. 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. From the same base FS checkpoint, we adapt 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 14). At 2k, S-NIAH-1 falls from 100 at $P = 64$ to 98 at $P = 32$ and 84 at $P = 16$, but accuracy is not monotone in P : at 16k, $P = 64/32/16$ score 74–80/66/68, using the two seed-42 controls at $P = 64$. 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 14: **Adapted retention budgets at 340M.** Each model trains and evaluates with its stated P .

Model	S-NIAH-1				S-NIAH-2				S-NIAH-3			
	2k	4k	8k	16k	2k	4k	8k	16k	2k	4k	8k	16k
$P = 16$	84	64	68	68	100	56	<u>24</u>	14	56	14	4	2
$P = 32$	<u>98</u>	<u>78</u>	<u>72</u>	66	96	<u>50</u>	32	<u>10</u>	74	20	<u>6</u>	2
$P = 64$, run 1	100	82	82	80	<u>98</u>	44	22	2	<u>66</u>	12	8	2
$P = 64$, run 2	100	76	70	<u>74</u>	96	46	20	2	58	<u>18</u>	8	2

Figure 8 relates these scores to the allocated attention buffers of the streaming cache (Table 15, protocol of Appendix C): reducing P fourfold gives a $2.4\times$ reduction in total buffers (which do not depend on length), while decode stays at 88–89 tokens/s. The $P = 64$ row of Table 15 uses the Table 1 checkpoint, while the $P = 64$ accuracies in Figure 8 average the two seed-42 continuations.

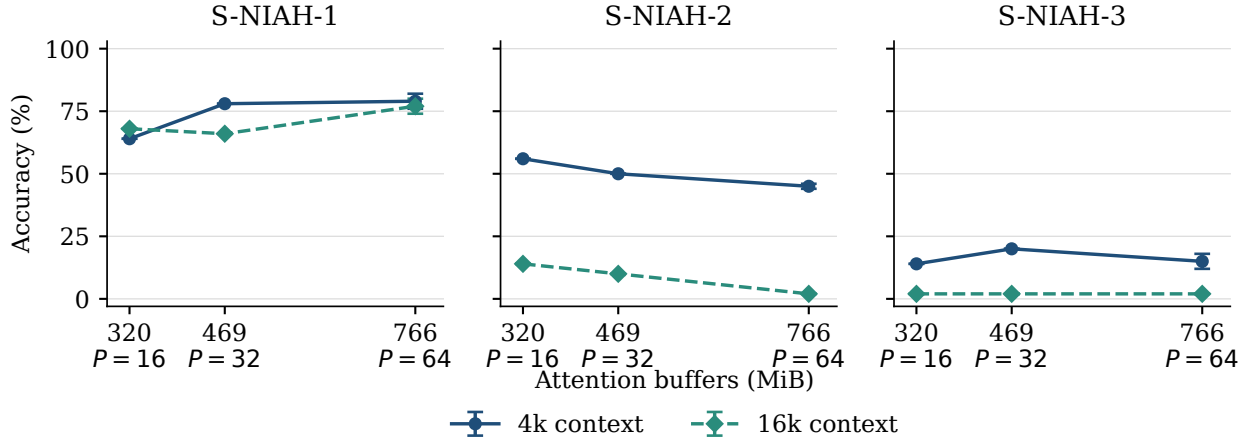


Figure 8: **Accuracy versus attention storage at 340M.** The $P = 64$ points show the mean and range of two seed-42 continuations.

Table 15: **Retention budget and bounded inference at 340M, 256k.**

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

Several facts at fixed context. Each synthetic prefix holds F distinct key–value facts with seven-digit values, each outside the final 528 prefix tokens. The question is revealed only after the prefix, and each question continues an independent copy of the prefix state. We generate 16 tokens greedily and score whether the target value appears, with 50 prefixes per condition, querying every fact for $F \leq 8$ and eight sampled facts for $F = 16$. Prefixes contain 4253–4277 tokens at 340M and 3946–3971 at 800M, where prefix, question and answer fit within 4096 tokens.

FS, FS-L, and FS-B pass a 90% single-fact threshold at both scales, but no model passes it with two or more facts (Table 16); FS-B scores highest at 340M; at 800M with four or more facts, full-attention TF++ and

Table 16: **Multiple-question retrieval.** Accuracy (%) on shared prefixes: an answer is correct if the target value appears in 16 greedy tokens; 50 prefixes per condition. Bold and underline mark the best and second-best score within each scale, including ties. FS-B prefills each question afresh; at 340M this raises FS and FS-L by about 2 points relative to the shared-prefix rows.

Facts per prefix	1	2	4	8	16
Questions	50	100	200	400	400
<i>340M, 4253–4277 prefix tokens</i>					
FS	<u>94</u>	41	19	7.75	2.75
FS-L	90	<u>48</u>	<u>21</u>	<u>8.00</u>	<u>3.25</u>
FS-B	100	56	28.5	13.0	4.25
<i>800M, strictly within 4096 tokens</i>					
FS	100	49	26	11.5	7.25
FS-L	<u>96</u>	53	21	10	4
TF++	88	56	38	22	19.75
Raven	100	<u>57</u>	28.5	9	2
FS-B	100	52	32	16.5	8.75
Raven-full	100	66	<u>33.5</u>	<u>21.75</u>	<u>13</u>

Raven-full lead instead. The decline also occurs with unpruned access, with full-attention TF++ and with Raven and Raven-full, and at 800M entirely within the training context, so it is not a pruning or length-extrapolation limit alone. At 800M with two facts, most wrong answers contain another value from the same prefix (42/47 for FS-L, 46/51 for FS, 48/48 for FS-B, 36/43 for Raven and 36/44 for TF++), consistent with confusion among stored facts. Because unpruned FS and full-attention TF++ decline in the same way, this test measures interference among facts near the training length rather than the retention limit above.

E FS-B details

Section 3.4 defines FS-B. This appendix explains how the forests are built and searched, what they cost, and the CPT recipe, and relates FS-B to chunk and hierarchical retrieval.

E.1 Streaming forests

Causal availability. Cells contain $s = 32$ tokens and query blocks contain $B = 128$ tokens. At each block boundary, cells completed before that boundary are admitted to the forests by Eq. (10). The forests and their summaries then stay fixed throughout the block, while each token searches independently. Prefill and cached decoding use the same block-boundary availability as FS training. Filtering a cell from the second forest does not remove it from the first, nor its tokens from the local window. The threshold τ_{ℓ_h} is fixed per checkpoint, so the second forest need not hold half of a particular prefix.

Chunks. Fix one head and one forest at a block boundary, and let $c_1, \dots, c_M$ be its admitted cells in arrival order, with mean keys $\bar{\mathbf{k}}_c$, key norms a_c and positions p_c (Eq. (5)). Write the binary decomposition $M = \sum_{t=1}^r 2^{k_t}$ with $k_1 > \dots > k_r$. The active chunks are the consecutive runs

$$\mathcal{G}_t = \{c_j : o_t < j \leq o_t + 2^{k_t}\}, \quad o_t = \sum_{u < t} 2^{k_u}, \quad (18)$$

so older cells sit in larger chunks; $13 = 8 + 4 + 1$ gives $\{c_1, \dots, c_8\}$, $\{c_9, \dots, c_{12}\}$ and $\{c_{13}\}$. A chunk never changes once complete, so its tree is built once and reused until two equal chunks merge into the next larger one.

Splitting a chunk. Each node v holds a cell set $\mathcal{C}(v)$ of size n_v ; the root of $\mathcal{G}_t$ holds all of $\mathcal{G}_t$. A node with $n_v > 1$ cells centers their mean keys, $\mathbf{x}_c = \bar{\mathbf{k}}_c - \boldsymbol{\mu}_v$, and estimates a principal direction with two power iterations started at the farthest cell:

$$\mathbf{u}^{(0)} \propto \mathbf{x}_{c^*}, \quad c^* = \arg \max_{c \in \mathcal{C}(v)} \|\mathbf{x}_c\|_2, \quad \mathbf{u}^{(t+1)} \propto \sum_{c \in \mathcal{C}(v)} (\mathbf{x}_c^\top \mathbf{u}^{(t)}) \mathbf{x}_c, \quad t = 0, 1. \quad (19)$$

Cells are sorted by $\mathbf{x}_c^\top \mathbf{u}^{(2)}$; the first $\lceil n_v/2 \rceil$ form the left child and the rest the right child. Recursion stops at single cells, so a chunk of 2^k cells gives a perfect binary tree of height k whose leaves are ordered by content rather than position.

Node summaries. Every chunk-tree node stores

$$\boldsymbol{\mu}_v = \frac{1}{n_v} \sum_{c \in \mathcal{C}(v)} \bar{\mathbf{k}}_c, \quad \rho_v = \max_{c \in \mathcal{C}(v)} \|\bar{\mathbf{k}}_c - \boldsymbol{\mu}_v\|_2, \quad A_v = \max_{c \in \mathcal{C}(v)} a_c, \quad p_v^{\min} = \min_{c \in \mathcal{C}(v)} p_c, \quad (20)$$

the quantities scored by Eq. (11).

Joining chunk roots. A chain joins the chunk roots from the smallest chunk upward: J_r is the root of $\mathcal{G}_r$, and for $t = r - 1, \dots, 1$ the join node J_t has children $\text{root}(\mathcal{G}_t)$ and J_{t+1} ; J_1 is the forest root. A join node v with children a, b stores

$$n_v = n_a + n_b, \quad \boldsymbol{\mu}_v = \frac{n_a \boldsymbol{\mu}_a + n_b \boldsymbol{\mu}_b}{n_v}, \quad \rho_v = \max_{u \in \{a, b\}} (\rho_u + \|\boldsymbol{\mu}_u - \boldsymbol{\mu}_v\|_2), \quad (21)$$

with $A_v = \max(A_a, A_b)$ and $p_v^{\min} = \min(p_a^{\min}, p_b^{\min})$. By the triangle inequality its ball encloses every cell mean below it, so Eq. (11) remains an upper bound at join nodes. The root of $\mathcal{G}_t$ sits at depth $\min(t, r - 1)$ and its tree has height $k_t \leq k_1 - t + 1$, so the forest height is $k_1 + [r \geq 2] = \lceil \log_2 M \rceil$.

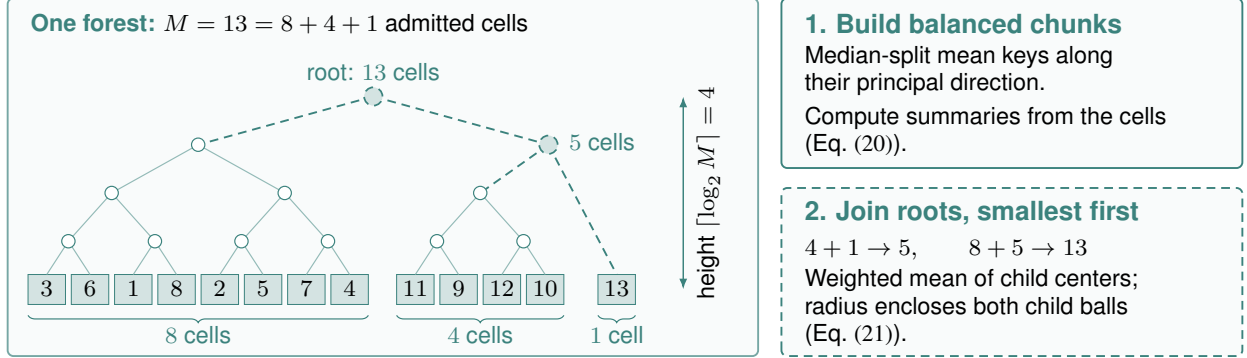


Figure 9: **Streaming construction of one FS-B forest, per head.** Solid edges form balanced trees within power-of-two chunks; dashed edges join their roots, smallest first. Leaf numbers denote arrival order, while their arrangement follows mean-key splits. The $13 = 8 + 4 + 1$ example has height $\lceil \log_2 13 \rceil = 4$. Right: the two construction steps; node summaries are defined in Eqs. (20)–(21).

Beam search and reranking. Search expands the current beam level by level and keeps the W^b highest-priority nodes under Eq. (11). A leaf reached early on a short branch keeps its slot, while the remaining slots expand children. The bound holds for the mean-key score of Eq. (7), not for the token-level score of Eq. (12) or for a cell’s usefulness to the answer; larger c favors subtrees with dispersed mean keys. We set $c = 8$ on S-NIAH at 341M and use it unchanged at 743M. The two candidate sets are merged without a quota for either forest. In Eq. (12), the log-sum-exp term is the logarithm of a cell’s unnormalized token-attention mass, and the prior affects selection only. Node summaries never replace token values.

Streaming inference. Prefill processes query blocks in fixed-size chunks. During decoding, the cache keeps only the active chunk trees. A completed larger chunk replaces its constituent smaller trees, and unchanged chunks are reused. For M admitted cells, these trees and their joining nodes contain fewer than $2M$ nodes, giving $O(Md)$ tree storage per head.

E.2 Cost

Let $M \leq \lfloor N/s \rfloor = \lfloor N/32 \rfloor$ be the number of admitted cells in a forest; the filtered forest can hold fewer cells than the full-prefix forest. For a balanced tree and $C_i = |\mathcal{B}_i|$ candidates, search, token-key reranking and readout take $O(d[W^b \log M + sC_i + sK])$ work per query, with $C_i \leq 2W^b$ for the two forests. Fixed budgets therefore give $O(N \log N)$ routing, reranking and token attention over the sequence, including the fixed local window.

Construction with radix sorting. Assume fixed-width keys for the split projections and stable radix sorting, so sorting m projections costs $O(m)$ work. Computing the principal direction, projections and summaries

at a node costs $O(md)$, hence a chunk of 2^k cells costs $O(d2^k k)$ over its k split levels. At most $\lfloor M/2^k \rfloor$ chunks of that size complete, and completed trees are reused. Summing over chunk sizes gives

$$O\left(\sum_{k=1}^{\lfloor \log_2 M \rfloor} \frac{M}{2^k} d2^k k\right) = O(Md \log^2 M). \quad (22)$$

Two forests change only the constant factor. Including search and readout, the algorithmic sequence cost is $O(N \log^2 N)$ at fixed dimensions and budgets, still strictly subquadratic. This bound assumes that updates reuse stored chunks without copying the growing forest at every block. Comparison sorting at every split would instead give construction work $O(Md \log^2 M + M \log^3 M)$; the stated bound uses fixed-width radix sorting or linear-time median partitioning.

Reference streaming overhead. The reference cache reuses completed trees but reallocates its growing admitted-cell index array and repacks all active node tables after updates. At fixed block size, these copies can accumulate $O(M^2 d)$ memory traffic over a growing decode stream, outside the algorithmic construction bound. Geometric buffer growth and persistent chunk storage can avoid this copying without changing tree contents or search. The measured reference throughput includes these overheads. The token archive uses $O(Nd)$ space and the active decode trees use $O(Md)$ storage.

E.3 Forest ablation

Table 17 compares FS-B’s two forests with single-forest ablations, changing only the inference-time candidate generator; the checkpoints and the four-cell token read are unchanged. At 800M, FS-B matches or exceeds the full-prefix width-32 ablation at every length, by up to 0.8 mean points at the displayed precision. The width-16 full-prefix ablation has the highest mean at 4k (37.9 versus 37.7 for FS-B). At 340M, the width-32 ablation leads the suite mean at all four lengths by up to 1.3 points. Its S-NIAH-2 scores match or exceed FS-B’s; its S-NIAH-3 scores are higher at 2k–8k, but lower at 16k (24 versus 27). At 800M, FS-B has higher FWE scores than the width-32 ablation at every length and higher MV-NIAH scores from 4k onward.

Two width-16 beams supply at most 32 distinct candidates after deduplication, so a width-32 single forest matches the maximum candidate count. A width-16 single forest has a smaller candidate budget and scores below width 32 at 8k–16k at both scales. Equal maximum candidate counts do not imply equal runtime. The norm-filtered width-16 ablation retains 41–74% of cells by task, using a median key-norm threshold calibrated on eight 4096-token FineWeb-Edu batches. Relative to FS-B, its suite mean is lower by 1.5–2.9 points at 340M and 0.2–1.5 points at 800M.

E.4 Retrieval-weighted CPT

Recall masks. Let f_t indicate a recall start or continuation, identified by a unique earlier four-token match more than 512 tokens away. A start matches the span beginning at the target token with a different preceding token; a continuation matches the preceding four tokens and their next target. Let χ_t mark a recall start within 64 tokens of the previous start whose source is more than 64 tokens from the previous source, which distinguishes switching to another span from returning to the same one. The masks respect document boundaries and do not depend on the model’s confidence. Only natural FineWeb-Edu text is used; benchmark prompts and answers are never training examples or sources of loss weights.

Table 17: **Forest ablation of FS-B** (0–100, RULER protocol of Table 1). All arms run on the same checkpoints with the same four-cell token read; only the inference-time candidate generator differs: FS-B’s two forests as rendered in Table 1, the full-prefix forest alone with a width-32 or width-16 beam, or the norm-filtered forest alone with a width-16 beam. Bold marks the best of the four arms per column.

Task	FS-B (two forests)				full, $W^b=32$				filtered, $W^b=16$				full, $W^b=16$			
	2k	4k	8k	16k	2k	4k	8k	16k	2k	4k	8k	16k	2k	4k	8k	16k
<i>≈340M scale</i>																
S-NIAH-1	100	99	99	99	100	99	99	99	99	99	99	99	100	99	99	99
S-NIAH-2	100	99	89	60	100	100	92	67	97	99	68	49	100	99	77	44
S-NIAH-3	91	78	64	27	95	84	71	24	75	61	51	22	90	79	70	26
MK-NIAH-1	27	25	19	24	27	28	23	21	29	24	22	19	30	20	23	19
MK-NIAH-2	2	0	0	1	2	0	0	1	2	0	0	1	2	0	0	0
MK-NIAH-3	2	0	0	0	1	0	0	0	1	1	0	0	1	0	0	0
MQ-NIAH	23.8	22.5	18.5	12.5	23.8	22.2	19.0	15.5	22.8	21.8	16.8	10.2	24.8	21.5	18.5	14.0
MV-NIAH	24.5	22.0	23.5	21.0	24.5	22.5	25.5	21.0	23.0	21.5	23.0	16.0	24.2	23.2	22.0	18.0
VT	3.2	1.2	1.0	1.0	3.0	1.4	1.6	1.2	2.8	1.2	1.0	1.2	3.0	1.2	1.0	1.4
CWE	5.2	10.3	6.5	1.8	6.7	10.0	6.6	2.0	5.0	10.6	6.6	1.8	5.3	10.4	7.3	1.3
FWE	9.0	6.0	11.7	5.0	9.3	4.7	9.0	5.7	8.7	3.7	6.7	3.0	9.7	6.0	9.7	6.7
QA-1	23	10	4	4	22	9	6	5	18	12	6	3	23	10	3	4
QA-2	–	–	9	6	–	–	10	6	–	–	8	7	–	–	7	6
Available-task mean	34.2	31.1	26.6	20.2	34.5	31.7	27.9	20.6	31.9	29.6	23.7	17.9	34.4	30.8	26.0	18.4
<i>≈800M scale</i>																
S-NIAH-1	97	98	97	100	95	98	99	99	97	99	100	99	97	99	98	100
S-NIAH-2	100	100	95	88	100	100	95	90	100	100	95	81	98	100	87	80
S-NIAH-3	96	93	87	84	96	93	87	87	88	88	84	81	93	94	90	82
MK-NIAH-1	50	34	28	31	46	25	30	29	52	28	28	30	48	28	27	26
MK-NIAH-2	9	4	1	2	5	4	1	2	8	3	1	3	11	1	2	2
MK-NIAH-3	2	1	0	0	2	0	1	0	1	0	0	0	2	0	0	0
MQ-NIAH	45.8	28.3	28.8	27.3	45.0	30.2	27.2	27.8	40.5	29.2	29.0	25.8	38.2	29.0	25.0	25.5
MV-NIAH	38.3	30.3	29.8	26.3	39.5	28.8	26.2	24.5	38.0	29.8	27.2	26.2	34.2	28.2	26.2	24.8
VT	3.8	1.4	1.0	0.6	4.0	2.0	0.8	0.4	4.8	2.0	0.8	0.2	4.0	2.4	1.0	0.6
CWE	7.8	26.4	8.9	4.4	7.8	29.2	8.1	3.4	10.0	24.9	9.9	3.2	9.6	30.1	7.4	2.3
FWE	17.7	8.7	7.0	8.0	15.3	7.3	3.3	5.7	10.3	4.7	7.7	15.0	23.0	10.7	7.3	8.3
QA-1	38	27	13	10	40	29	14	13	39	26	13	11	40	32	14	12
QA-2	–	–	17	14	–	–	18	14	–	–	15	9	–	–	14	11
Available-task mean	42.1	37.7	31.8	30.4	41.3	37.2	31.6	30.4	40.7	36.2	31.6	29.6	41.5	37.9	30.7	28.8

Schedule. Starting from the base 800M FS checkpoint, CPT has two 800-step stages of 262,144 tokens per step with $\beta = 9$; β_{sw} is 9 in the first stage and 49 in the second, so a switch token has weight 19 and then 59. In each stage the learning rate warms up for 20 steps to 4×10^{-5} and decays to 4×10^{-6} . The total is 1600 steps and 0.42B tokens, all at 4096 tokens per sequence, so evaluation beyond 4k remains zero-shot extrapolation. During CPT, FS scores the eligible prefix and reads 16 cells per head; at evaluation, FS-B searches the two forests and reads four. The forests are not in the training loop, and no routing parameters are learned.

CPT ablation with single-forest readout. Table 18 compares the base and CPT checkpoints under the single-forest ablation read paths, holding each selector fixed: the 0.42B-token continuation improves every arm and length, with the largest gains at 16k. The gains concentrate on the retrieval tasks that the weighting targets (S-NIAH-1/3 improve by 4–18 points and multi-key and multi-value retrieval by 4–8), with small regressions on CWE of up to 7.8 points and on FWE of up to 3.7. Reading the same CPT checkpoint with FS-B’s two-forest search gives 42.1/37.7/31.8/30.4 (Table 1), exceeding the full-prefix width-32 ablation at 2k–8k and matching it at 16k at the displayed precision. The per-task breakdown of the CPT arms is in Table 17. The pools store aggregate scores per task, so no prompt-level intervals are reported.

Table 18: **Recall-weighted CPT under single-forest ablations** (0–100; RULER available-task means over 12 tasks at 2k/4k and 13 at 8k/16k, protocol of Table 1). All rows start from the base 743M FS checkpoint; CPT rows are the Table 1 FS-B checkpoint evaluated with a single-forest ablation. FS-B denotes the two-forest design, whose results appear in Table 1.

Read path	Weights	2k	4k	8k	16k	Mean
Full-prefix forest (W32)	base FS	40.9	36.0	30.4	28.2	33.9
	+weighted CPT	41.3	37.2	31.6	30.4	35.1
Norm-filtered forest (W16)	base FS	39.6	35.1	29.8	27.5	33.0
	+weighted CPT	40.7	36.2	31.6	29.6	34.5

E.5 Relation to chunk and hierarchical retrieval

Several methods retrieve distant chunks. HCAM (Lampinen et al., 2021) and the Hierarchical Sparse Attention of RAMba (Hu et al., 2025a) score chunk summaries, attend within selected chunks, and combine the chunk outputs; DRT’s Grouped Cross-Attention (Hu et al., 2025b) retrieves past chunks for each next chunk. Their routing scores weight the chunk outputs, which gives the router a gradient, and DRT and RAMba offload growing token memory to the CPU while keeping chunk keys on the GPU. FreeSolo instead uses hard cell selection followed by one softmax over the selected tokens, so its shared projections learn through the token readout and local attention; FS-L bounds the accessible cells by $P + W$ per head, and FS-B searches two content forests that together cover every cell. HMT (He et al., 2025) recalls compressed segment memories within a recurrent architecture, whereas FreeSolo keeps the token-level keys and values of every retrievable cell. Treeformer’s TF-attention (Madaan et al., 2023) learns decision-tree partitions and attends to keys routed to the query’s leaf; FS-B builds balanced partitions from the mean keys of causally available cells and searches several branches with query-dependent priorities. The published protocols also differ: DRT uses task fine-tuning and RAMba’s RULER tests include synthetic fine-tuning at 4k, whereas our models use next-token pretraining only, so we do not compare published scores.