Sparse Delta Memory

Scaling the State of Linear RNNs through Sparsity

Paper by Loïc Cabannes et al.

Independent paper-reading presentation

arXiv:2607.07386v1

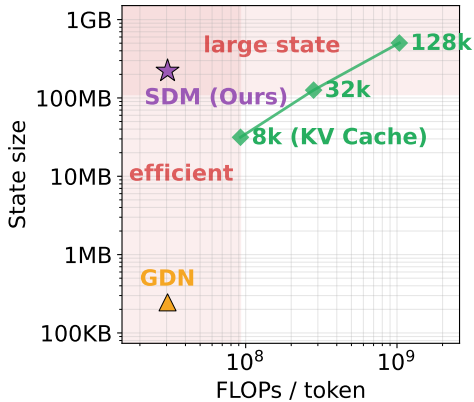
8 September 2026 · 20–30 minute group discussion

A larger recurrent memory improves recall when only a few slots are accessed.

Can recurrent memory grow without dense updates?

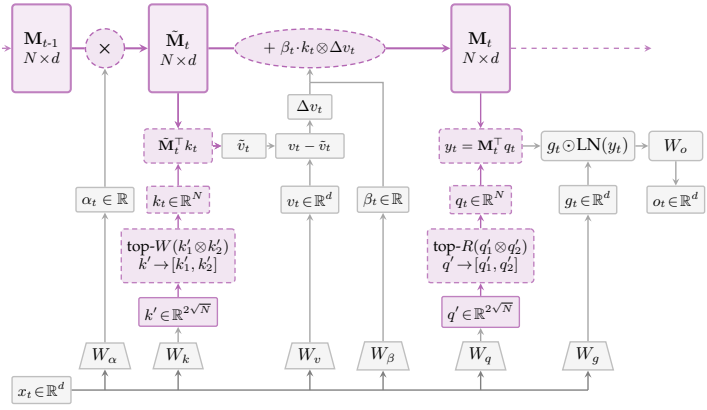
- Full attention retains a growing KV cache.
- GDN keeps a compact state but touches it densely.
- SDM increases stored capacity while limiting the slots accessed per token.

Capacity N and access budgets W, R become separate controls.



Paper, Secs. 1–2, pp. 1–3; original Fig. 1, p. 1. Plot is state vs FLOPs, not measured throughput.

SDM routes writes and reads into a large explicit table



Purple/dashed: sparse operations. Unselected write slots remain unchanged.

Original Fig. 2, PDF p. 3, cropped from the final paper. The printed update and this diagram differ; see slide 6.

Product keys select addresses without scoring every slot

$$\mathbf{k}'_t = [\mathbf{k}'_{1,t}, \mathbf{k}'_{2,t}] \in \mathbb{R}^{2\sqrt{N}}, \qquad s_{a,b} = k'_{1,t}[a] + k'_{2,t}[b]$$

1. Project the input into two score vectors of length $\sqrt{N}$.
2. Keep each half's top-k; form only k^2 candidate sums.
3. Select top-W write addresses and top-R read addresses.
4. Apply softmax over selected scores to obtain sparse weights.

Addressing cost

$O(\sqrt{N}\, d + W^2 + R^2)$; no need to materialize all N scores.

Paper, Secs. 2 and 3.1, pp. 3–4. The text specifies an outer sum; Fig. 2 uses an inconsistent product symbol.

Selected slots decay and update; other slots persist

For each selected write slot $i \in \mathcal{I}_t^w$, v1 prints:

$$\widetilde{\mathbf{M}}_t[i] = \alpha_t \mathbf{M}_{t-1}[i], \qquad \mathbf{M}_t[i] = \widetilde{\mathbf{M}}_t[i] + \beta_t k_t^{(i)} (\mathbf{v}_t - \widetilde{\mathbf{M}}_t[i]).$$

$$i \notin \mathcal{I}_t^w : \ \mathbf{M}_t[i] = \mathbf{M}_{t-1}[i], \qquad \mathbf{y}_t = \sum_{i \in \mathcal{I}_t^r} q_t^{(i)} \mathbf{M}_t[i].$$

- α_t: per-head forget gate; β_t: input gate.
- $k_t^{(i)}, q_t^{(i)}$: sparse write/read weights; $\mathbf{v}_t$: value vector.
- Read after writing, then RMS-Norm, output gating, and head mixing.

Paper, Sec. 3.1, p. 4, Eqs. (3)–(5). This is Eq. (4) as printed; its residual differs from Fig. 2.

A v1 inconsistency matters for reproducing the update

Fig. 2 reads one shared prediction before writing

$$\widehat{\mathbf{v}}_t = \widetilde{\mathbf{M}}_t^{\top}\mathbf{k}_t, \qquad \mathbf{M}_t[i] = \widetilde{\mathbf{M}}_t[i] + \beta_t k_t^{(i)}(\mathbf{v}_t - \widehat{\mathbf{v}}_t).$$

- Eq. (4) instead subtracts the individual slot $\widetilde{\mathbf{M}}_t[i]$.
- The shared residual matches GDN and the key interactions in Appendix A.
- These updates differ for general multi-slot keys.

The printed Eq. (4) does not establish the claimed exact dense GDN limit.

Paper, Eq. (2), p. 2; Fig. 2, p. 3; Eq. (4) and Connection to GDN, p. 4; App. A, pp. 14–15. Code not audited.

Head count controls a cubic state expansion

$$N = \left(\frac{d}{4H}\right)^2, \qquad d_{\mathrm{v}} = \frac{d}{H}, \qquad M_{\mathrm{size}} = HNd_{\mathrm{v}} = \frac{d^3}{16H^2}.$$

1.4B configuration

$d = 1920,\ H = 2$
$N = 57{,}600$ slots per head
553M state elements over 5 global layers
$\approx 900\times$ GDN

8B configuration

$d = 3840,\ H = 2$
$N = 230{,}400$ slots per head
7.963B state elements over 9 global layers
$\approx 3600\times$ GDN

Paper, Sec. 3.3, Eq. (6), p. 5; Table 1, p. 6. Slot counts and ratios calculated from configurations. Elements are not bytes.

Matched FLOPs exclude the cost of storing more parameters

- Match projection widths: $d_{\mathrm{qk}}^{\mathrm{total}} = d/2$ and $d_{\mathrm{v}}^{\mathrm{total}} = d$.
- Set $W = R = 64$; the paper reports extra PKM selection below 1% of layer FLOPs.
- Default SDM learns $\mathbf{M}_0$, but the reported parameter count excludes this memory.

Scale	Reported params	Learned state	Sum (derived)
1.4B	1.48B	0.553B	2.033B
8B	8.14B	7.963B	16.103B

This is not a comparison at equal total learnable storage.

Paper, Sec. 3.2, pp. 4–5; Table 1 and Sec. 5.1, p. 6. Sums use rounded reported counts.

Training parallelizes within chunks and recurs across chunks

Batched work within chunks

- Accumulate decay per accessed slot.
- Build sparse write–write and read–write interactions.
- Solve a triangular system for delta corrections.

Sequential work across chunks

- Gather the incoming memory.
- Correct, then scatter updates.
- Combine past-state reads with within-chunk contributions.

Backward undoes sparse writes instead of saving a full table at every chunk.

Paper, App. A.1–A.3, pp. 14–15. This describes the appendix's shared-residual mechanism; the v1 formula caveat remains.

The comparisons use a shared local/global hybrid

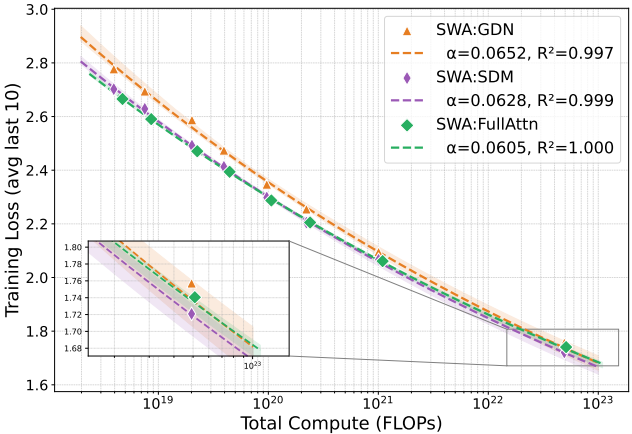
Architecture

SWA (window 128) and global layers interleaved at roughly 3:1.
Replace the global layer with FullAttn, GDN, Mamba2, or SDM.

- Pretrain at 8192 tokens; AdamW and a warmup–stable–decay schedule.
- Scale from 257M to 8.14B counted parameters at 160 tokens per non-embedding parameter.
- 1.4B: 168.4B pretraining tokens, then 4B tokens at 128k.
- 8B: 1.141T pretraining tokens, then 16B tokens at 128k.

Paper, Sec. 4, pp. 5–6; Table 1, p. 6. FullAttn in the results is also a SWA/global hybrid.

SDM achieves lower training loss than GDN across the ladder



- Fit levels 1–8; predict level 13.
- SDM fit: $R^2 = 0.999$.
- At 8B, SDM also beats FullAttn in training loss.

This fit does not establish benchmark significance.

Paper, Sec. 5.1, p. 6; original Fig. 3, p. 7. Loss averages the last 10 log entries. Memory parameters excluded from matching.

Short-context averages improve, with substantial task variation

Scale	Metric	FullAttn	GDN	SDM
1.4B	Text NLL ↓	2.660	2.683	**2.654**
	Average score (%) ↑	37.98	37.93	**38.51**
8B	Text NLL ↓	2.285	2.298	**2.253**
	Average score (%) ↑	56.65	55.70	**56.84**
	BoolQ (%) ↑	71.83	**74.98**	67.13

- 8B: +1.14 percentage points vs GDN; +0.19 vs FullAttn.
- BoolQ drops 7.85 points vs GDN. No benchmark error bars reported.

Paper, Table 2, p. 8; post-trained models, 15-task average. 8B NLL difference recalculates to -0.045, not the printed -0.040.

RULER improves over GDN; FullAttn still leads overall

Scale	FullAttn	GDN	SDM	SDM minus GDN
1.4B	**32.5**	20.0	31.2	+11.2 pp
8B	**61.2**	34.2	50.2	+16.0 pp

Protocol

Exact-match accuracy (%), averaged over 13 tasks and lengths 4k–131k, after 128k fine-tuning.

- 8B SDM remains 11.0 points below FullAttn.
- The body also reports 76.2 for FullAttn under an inconsistent task discussion; this slide uses Table 2.

Paper, Sec. 5.3, pp. 6–7; Table 2, p. 8. pp = percentage points.

Hard multi-key retrieval remains a clear weakness

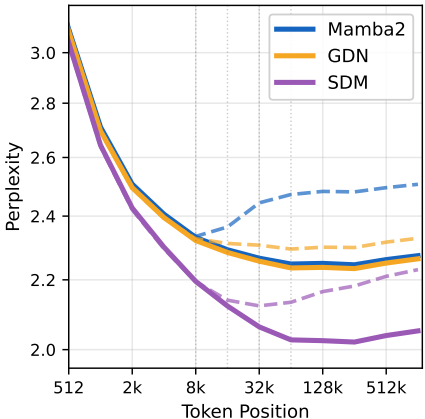
8B RULER task, accuracy (%)	FullAttn	GDN	SDM
single_3	70.1	32.6	**74.9**
multiquery	**73.0**	31.2	68.6
variable tracking	67.2	46.2	**72.3**
multikey_2	**58.0**	1.0	4.7
multikey_3	**29.8**	0.1	1.2

SDM improves or matches GDN on all 13 tasks at 8B.

At 1.4B, however, frequent-word extraction falls from 35.6 to 12.3.

Paper, Table 2, p. 8. Selected task rows; averages span lengths. Full length curves: Figs. 9–10, p. 17.

More context improves code modeling into very long documents



- 1.4B models; 1M-token code documents.
- Solid: 128k fine-tuned. Dashed: pretrained only.
- SDM approaches PPL 2.0; dense RNNs stay around 2.2–2.3.
- Raw PPL rises after 256k. Authors attribute this to local token difficulty.

Paper, Sec. 5.4, p. 7; original Fig. 4, p. 9. Values are approximate. The claimed isolated global-layer gain is not plotted here.

Capacity helps recall even without a learned initial memory

1.4B model	$\mathbf{M}_0$	Code NLL ↓	Avg. score (%)	RULER (%)
GDN	zero	0.849	37.9	20.0
GDN	learned	0.850	38.0	20.6
SDM	zero	0.845	37.3	28.0
SDM	learned	**0.822**	**38.5**	**31.2**

- Zero-initialized SDM gains 8.0 RULER points over zero-initialized GDN.
- Learning SDM's initial table adds another 3.2 points and lowers code NLL.
- Learned $\mathbf{M}_0$ supplies pretrained information; context updates modify $\mathbf{M}_t$ during inference.

Paper, Sec. 6, pp. 7–9; Table 3 and Fig. 5, p. 9. Accuracy columns: higher is better.

Shrinking the memory gradually degrades NLL and recall

0.8B model	State/layer	Code NLL	Avg. score (%)	RULER (%)
GDN, zero $\mathbf{M}_0$	0.2 MB	0.963	33.8	16.0
SDM, learned $\mathbf{M}_0$	27 MB	0.947	33.7	20.2
SDM, learned $\mathbf{M}_0$	108 MB	0.937	33.6	20.7
SDM, learned $\mathbf{M}_0$	432 MB	**0.914**	**34.6**	**21.5**

- Larger state improves code NLL (↓) and RULER (↑) monotonically here.
- Short-task averages are not strictly monotonic.
- Varying the key projection width changes N; this is not a pure byte-capacity intervention.

Paper, Sec. 6, pp. 7–8; Table 3 (bottom), p. 9. MB labels retained from paper; W, R, d_v and non-SDM layers fixed.

Sparse arithmetic still pays for random memory access

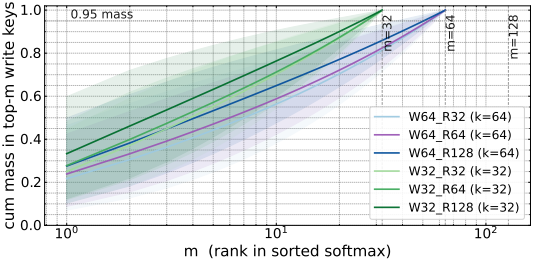
The hardware bottleneck

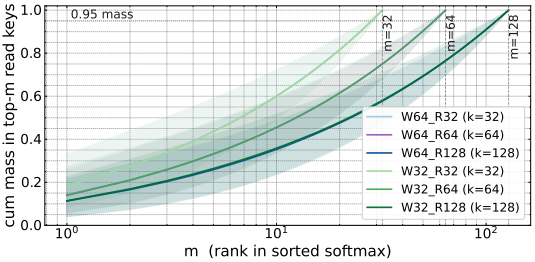
GDN's small state fits in SRAM. SDM's large table resides in HBM.

- Current SDM kernel MFU is about one order of magnitude below the optimized GDN kernel.
- 8B end-to-end training: reported as 1.49× slower than GDN.
- 1.4B decode: about 10% slower than GDN, about 6× faster than FullAttn in the reported test.
- 8B SDM state: as much memory as 203,400 tokens of the compared FullAttn KV cache.

Paper, Sec. 6, pp. 8–9, and Sec. 7, p. 10. Full batch/length timing protocol is not reported; these are not universal speed ratios.

Read weights spread more broadly than write weights





Writes: top 32 of 64 carry $\sim 85\%$.

Reads: top 64 of 128 carry $\sim 77\%$.

$$C(m) = \sum_{i=1}^{m} p_{(i)}, \qquad p_{(1)} \geq p_{(2)} \geq \cdots$$

Curves: mean cumulative mass; shading: token-level p10–p90, not training-run confidence intervals.

Paper, Sec. 6, p. 9; original Fig. 6(a,b), p. 10; App. D.1, p. 18. Approximate coverage depends on configuration.

Full attention and SDM show complementary behavior

1.4B pretrained	SWA:FA	SWA:(FA/SDM)	SWA:SDM
Text NLL ↓	2.658	**2.645**	2.654
Code NLL ↓	0.798	**0.797**	0.821
Average score (%) ↑	38.1	**38.6**	**38.6**
RULER (%) ↑	**80.1**	64.3	56.9

- Alternate global layers: FA–SDM–FA–SDM–FA.
- RULER here covers only 6 tasks at 4k–8k; do not compare it directly with the 13-task main table.
- The mixed model retains a growing global KV cache.

Paper, App. E, p. 19; Table 5, p. 20. Fig. 11 fits levels 1–8; it does not report a trained 8B mixed model.

The evidence supports a useful tradeoff, with clear boundaries

- **Capacity:** larger fixed state helps, but difficult multi-key tasks remain weak.
- **Comparison:** parameter matching excludes learned memory; matched FLOPs do not imply equal wall time.
- **Reproducibility:** the update equation and several numerical statements need clarification in v1.
- **Scope:** text/code experiments reach 8B and million-position code modeling; long-lived agents and video memory remain future directions.

Paper, Secs. 5–7, pp. 6–10; App. A–F, pp. 14–20. Interpretation based on reported experiments; no independent reproduction.

Three points to retain

1. **Scale capacity through sparse access.**
   A large recurrent table can be useful without updating every element at each token.
2. **Separate dynamic state from learned initialization.**
   Both contribute, and they answer different memory questions.
3. **Evaluate storage and real speed alongside FLOPs.**
   The next bottleneck is maintaining and accessing a large per-sequence state.

Reading synthesis. Main evidence: Tables 1–3, pp. 6, 8–9; efficiency and limitations, pp. 8–10.

Paper identity and materials used

Sparse Delta Memory: Scaling the State of Linear RNNs through Sparsity

Loïc Cabannes, Pierre-Emmanuel Mazaré, Gergely Szilvasy, Matthijs Douze, Maria Lomeli, Ilze Amanda Auzina, Justin Carpentier, Gabriel Synnaeve, Hervé Jégou.

arXiv:2607.07386v1, submitted 8 July 2026.

- Read same-version author TeX, final PDF, and original figures.
- Covered main text and Appendices A–F.
- All experimental figures are original; selected table rows preserve paper values.
- This presentation is a reading analysis; authors' code was not run or audited.

Retrieved 8 September 2026. PDF cover date is 9 July 2026. Local companion: reading-notes.zh.md; figure provenance: figures/manifest.json.

Backup: two slots expose the update discrepancy

$$\alpha = 0.8, \quad \beta = 0.5, \quad \mathbf{M}_{t-1} = [2.5,\ 5]^\top, \quad \widetilde{\mathbf{M}}_t = [2,\ 4]^\top, \quad \mathbf{k} = [0.5,\ 0.5]^\top, \quad v = 10.$$

Eq. (4): individual slot residual

$$[2,\ 4] + 0.25\,[10 - 2,\ 10 - 4] = [4,\ 5.5].$$

Fig. 2: shared prediction residual

$$\widehat{v} = 0.5(2) + 0.5(4) = 3.$$
$$[2,\ 4] + 0.25\,[10 - 3,\ 10 - 3] = [3.75,\ 5.75].$$

Teaching counterexample constructed for this reading; not experimental data. Compare Eq. (4), p. 4, with Fig. 2, p. 3.

Backup: sparse interactions drive the triangular solve

$$\mathbf{A}[i,j] = \sum_{n,m:\ \mathcal{I}_i^w[n]=\mathcal{I}_j^w[m]} k_i^{(n)} k_j^{(m)} e^{\lambda_{i,n}-\lambda_{j,m}}, \qquad j < i.$$

- i, j: tokens within a chunk; n, m: positions in their selected-slot lists.
- λ: cumulative log-decay for each accessed slot.
- Sorted indices allow an $O(W)$ two-pointer merge per write–write pair.

$$\mathbf{M}_{\mathrm{sys}} \Delta\mathbf{V}_{\mathrm{const}} = \mathrm{diag}(\beta)\mathbf{V}, \qquad \mathbf{B} = \mathbf{M}_{\mathrm{sys}}^{-1}\,\mathrm{diag}(-\beta).$$

$\mathbf{V}$ stacks values; $\mathbf{M}_{\mathrm{sys}}$ has unit diagonal and $\beta_i \mathbf{A}[i,j]$ below it. $\mathbf{B}$ corrects incoming-state reads.

Paper, App. A.1–A.2, pp. 14–15; Eq. (7), p. 15. Interaction construction still costs $O(C^2 W)$ per chunk of length C.

Backup: undoing writes avoids repeated full-table snapshots

$$O\left(\frac{T}{C} N d_{\mathrm{v}}\right) \quad \longrightarrow \quad O(N d_{\mathrm{v}} + T W d_{\mathrm{v}}) .$$

- Forward: update the table in place and retain sparse update information.
- Backward: undo updates to reconstruct the required historical state.
- Reported example: 8B, $T = 16384$, $C = 128$, $W = 64$.

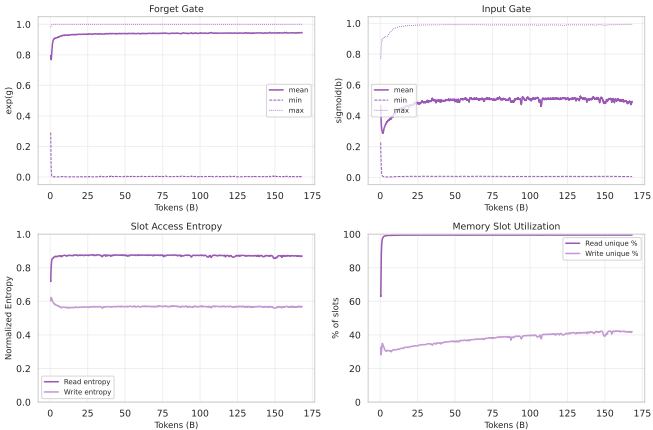
State checkpoint memory per SDM layer

226 GB $\longrightarrow$ 8 GB in the paper's example.

Authors also report fp8/int4 snapshots without observed quality loss, but give no complete quantitative comparison.

Paper, App. A.3, p. 15. Counts concern state checkpoints, not total training GPU memory.

Backup: aggregate slot coverage is not per-token coverage



- Mean forget gate ~ 0.95.
- Mean input gate ~ 0.50.
- Reads cover $\sim 100\%$ of slots over the probe window.
- Writes grow from $\sim 28\%$ to $\sim 42\%$.

Original Fig. 8 and App. B, p. 16. 1.4B; each point pools 10 training steps ($\sim$21M tokens), averaged across 5 SDM layers.

Backup: increasing the read budget does not always improve recall

1.4B pretrained	DCLM NLL ↓	RULER (%)	Reasoning (%)
W64 / R128	**2.6703**	40.11	38.42
W64 / R64	2.6719	**41.76**	38.68
W32 / R64	2.6726	35.01	**38.90**
SWA:FullAttn	2.6729	34.74	38.05
W32 / R128	2.6737	40.39	38.21
W64 / R32	2.6746	37.64	38.87
W32 / R32	2.6766	36.12	38.71

RULER here averages 6 tasks. The paper does not fully specify length aggregation for this table.

Effective selected-key utilization: $u(p) = \exp(-\sum_i p_i \log p_i)/k$.

Paper, Table 4, p. 19; App. D.1, p. 18. Scores converted from fractions to percent; larger accuracy is better.

Backup: numerical and protocol checks for v1

- **Short tasks:** Sec. 5.2 claims 13/15 wins at 1.4B; Table 2 contains 11 improvements and 4 regressions vs GDN.
- **8B text NLL:** $2.253 - 2.298 = -0.045$; the printed delta is -0.040.
- **RULER:** the body gives FullAttn 76.2; the 13-task main table gives 61.2.
- **Access probe:** App. D.1 says both truncation to 2048 and a batch of 8×8192 tokens.

Use the explicitly labeled tables; preserve unresolved differences when reproducing.

Paper, Secs. 5.2–5.3, pp. 6–7; Table 2, p. 8; App. D.1, p. 18. Checked in author TeX and final PDF.