A HIPPOCAMPUS FOR LINEAR ATTENTION

AN EXACT MEMORY FOR WHAT THE RECURRENT STATE FORGETS

Anonymous authors
Paper under double-blind review

ABSTRACT

Linear-attention and state-space language models compress the prefix into a fixed-size recurrent state, which makes them efficient but leaky: once many key–value associations compete, earlier facts are overwritten and exact recall of distant tokens fails. The brain faces the same trade-off and resolves it with complementary learning systems: a slowly consolidating neocortex compresses experience, while a fast hippocampal store keeps the surprising episodes the neocortex cannot absorb. Many efficient hybrids already give the recurrent state such a store — a small softmax-attention cache that holds exact key–value pairs — but fill it with the most recent tokens, as if the state could not tell what it has lost. We show that it can: in a delta-rule model, the magnitude of the update a token commits to the state measures exactly how poorly that token fits the compressed memory, and it is already computed at every step. HOLA (Hippocampal Linear Attention) adds to every Gated DeltaNet layer a small exact key–value memory that keeps the top-scoring tokens by this signal, however far back they are. The mechanism is visible directly: on a 32k-token sequence, the trained model's 64-slot memory still holds the needle 29k tokens later, whereas a recency cache of the same size has long dropped it. At 1B parameters trained on 50B tokens, HOLA surpasses published Gated DeltaNet, KDA, and their preconditioned variants on in-context retrieval (four-task average 35.4 vs. at most 32.7) while staying on par with them in language modeling and commonsense reasoning.

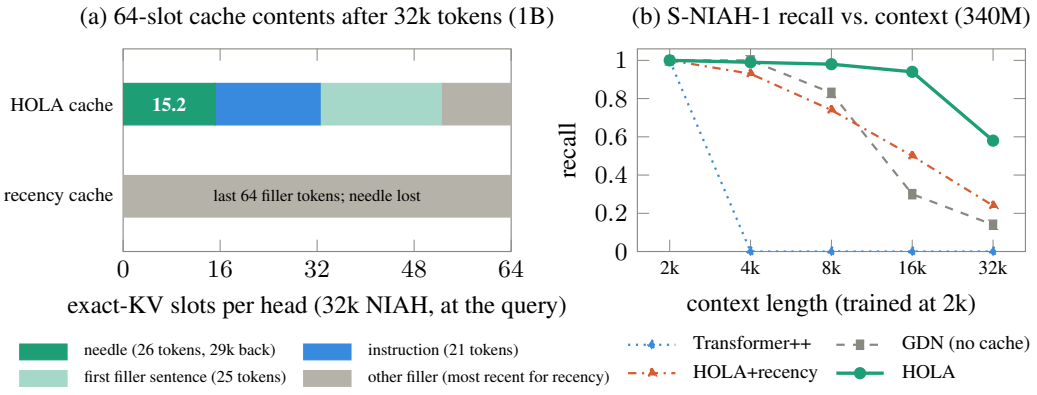


Figure 1: **HOLA keeps the surprising tokens, not the recent ones, and recalls them at length.** **(a)** Contents of the trained 1B model's 64-slot exact cache (per head, deep layers) at the query of a 32k needle-in-a-haystack sequence: 15.2 of the 26 needle tokens are still cached 29k tokens later, together with the instruction and the first occurrence of the repeated filler sentence; a recency cache of the same size holds only the last 64 filler tokens. **(b)** On RULER S-NIAH-1 at 340M, where we train matched controls, HOLA remains stronger than GDN and HOLA+recency as context grows to 32k tokens.

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

1 INTRODUCTION

The human brain does not rely on a single memory. A hippocampus-centered system can record a specific, novel event in one shot and later recall it precisely (Tse et al., 2007), while a neocortex-centered system slowly distills generalizable structure across many experiences (Frankland & Bontempi, 2005); Complementary Learning Systems (CLS) theory (McClelland et al., 1995; Kumaran et al., 2016) argues that the two *must* be separate — a system built for slow, compressive generalization would suffer *catastrophic interference* if forced to write single facts quickly, and so cannot also serve fast, exact, one-shot memory. Efficient language models are missing exactly this complementarity.

Linear attention is precisely such a "neocortex": a superb compressor, but a leaky exact memory. DeltaNet, Gated DeltaNet (GDN), and related linear-attention / state-space models (Katharopoulos et al., 2020; Gu & Dao, 2024; Yang et al., 2025; 2024b) replace softmax attention's growing key–value cache with a fixed-size recurrent state S, summarizing all history in $O(1)$ memory and processing a sequence in sub-quadratic time. The state is updated online — the *delta rule* writes the residual between each new value and the state's current prediction — making it an efficient running summary. But a fixed-size state is a *lossy* memory: it holds only a bounded number of distinct key→value associations before new writes overwrite old ones along shared key directions. This shows up exactly where it hurts: multi-item associative recall (Arora et al., 2024b), passkey / needle-in-a-haystack retrieval (Mohtashami & Jaggi, 2023), and copying a specific distant token verbatim (Jelassi et al., 2024). There, full softmax attention — which keeps every token exactly, at $O(T)$ memory and $O(T^2)$ compute — remains the gold standard. The question becomes: can we keep linear attention's cost while recovering its lost exact recall? CLS points to the answer — give this neocortex a hippocampus: a separate, exact, budget-limited episodic memory that preferentially stores *surprise*. The central design question for such a bounded exact memory is *what to store*.

What to store: a surprise signal the model already computes. A budget-limited exact memory is only useful if it keeps the *right* tokens. Most cache-bounded hybrids keep the most *recent* ones (Du et al., 2026; Wang et al., 2025; Fang et al., 2025), like StreamingLLM's sliding window for Transformer KV caches (Xiao et al., 2024); H2O (Zhang et al., 2023) keeps the tokens with the most accumulated attention. Neither asks whether the state failed to absorb a token, and recency cannot hold a fact that is far away yet must be recalled exactly. Our observation is that a delta-rule model already measures how surprising each token is to its state: the magnitude of the update it commits, $\beta \cdot \|e\|$. We use this intrinsic, parameter-free quantity as the cache's eviction score, keeping the tokens that changed the state most rather than the most recent ones. Figure 1a shows the effect on a real 32k sequence: the trained model's cache still holds the needle 29k tokens later, together with other first occurrences, where a recency cache of the same size holds only the last 64 filler tokens.

Together, these ideas give **Hippocampal Linear Attention (HOLA)**, which recovers much of the exact recall the state loses (Fig. 1). At 1B parameters trained on 50B tokens, HOLA surpasses the published Gated DeltaNet, KDA, and preconditioned (PGDN, PKDA) baselines on in-context retrieval by a clear margin while staying on par in language modeling and commonsense; at 340M, where we train matched controls, it lifts same-backbone in-context retrieval and remains far more robust than GDN on RULER single-needle recall out to 32k tokens ($16\times$ its 2k training length), where the recurrent state saturates and GDN decays.

Contributions.

1. **A hippocampal exact memory for linear attention.** Motivated by CLS, we add a bounded exact KV store to every delta-rule layer. It writes by an intrinsic, parameter-free surprise signal the delta rule already computes — the committed update magnitude $\beta \cdot \|e\|$ — and reads through a decoupled RMSNorm-γ path that makes exact KV pairs retrievable rather than softly averaged, at about 0.001% extra parameters and $O(1)$ memory.
2. **A mechanism-level account of what the cache does.** On real sequences we show that the cache keeps first occurrences regardless of distance and reads them sharply (Fig. 1a, Fig. 3); it also carries article context across document-boundary resets of the recurrent state, which explains why its perplexity gain depends on the evaluation protocol (App. G).
3. **Retrieval gains where the state fails.** At 1B/50B, HOLA outperforms all published linear baselines we compare on in-context retrieval (four-task average 35.4 vs. at most 32.7; SWDE

54.2 vs. 50.3). At 340M, against same-backbone controls, it raises in-context retrieval from 22.8 (GDN) and 23.7 (recency cache) to 27.8, and on RULER single-needle recall at 32k it holds 0.58 where GDN falls to 0.14 and the recency cache to 0.24. Language modeling and commonsense stay on par, at 3.3% more peak decode memory than GDN at 1B.

2 Related Work

Our work connects three threads: linear-time recurrent attention, hybrid exact-memory mechanisms for efficient language models, and semiparametric memory.

Linear-time recurrent attention. Linear attention (Katharopoulos et al., 2020), state-space models (Gu & Dao, 2024; Dao & Gu, 2024), and the DeltaNet family (Schlag et al., 2021; Yang et al., 2024b) replace softmax attention's growing KV cache with a fixed-size recurrent state. Gated DeltaNet (GDN) (Yang et al., 2025) strengthens DeltaNet with data-dependent decay, and GLA (Yang et al., 2024a), GSA (Zhang et al., 2024), and Kimi Delta Attention (KDA) (Kimi Team et al., 2025) are representative modern linear-attention baselines. This fixed state is efficient but lossy: as an associative memory, it has bounded capacity before new writes interfere with old ones (Hopfield, 1982; Ramsauer et al., 2021). Based and Zoology (Arora et al., 2024a;b) make this recall–throughput trade-off explicit, showing that efficient recurrent models lag softmax attention on multi-item associative recall. Titans (Behrouz et al., 2025) writes to a neural long-term memory with test-time gradient steps it calls surprise, which for a linear memory reduce to the delta-rule write whose magnitude is our score; it uses surprise to decide how strongly to write into the compressive memory, whereas HOLA uses it to decide what to keep exactly outside it. HOLA keeps the recurrent backbone, but adds a bounded exact memory for the KV pairs the state should not be forced to compress.

Hybrid exact memory for efficient LMs. Many efficient LMs recover some exact recall by pairing a recurrent/SSM backbone with softmax attention. Inter-layer (or layer-wise) hybrids such as Jamba (Lieber et al., 2024), Samba (Ren et al., 2025), and Griffin (De et al., 2024) periodically interleave recurrent layers with global or local softmax-attention layers. Global variants preserve exact access to the full prefix but retain a length-growing KV cache in those layers, whereas local variants bound this cost through a recency window. Layer-internal hybrids such as NHA (Du et al., 2026), RAttention (Wang et al., 2025), and AHN (Fang et al., 2025) instead combine recurrent memory with a recent-token window or learned compressed store inside a layer. In the cache-bounded variants, the lossless component is primarily recency-based: once an old token leaves the window, it must be represented by the compressed state. HOLA instead makes the exact memory selective over the prefix, retaining surprising KV pairs even when they are far from the current query.

The closest work is LTE (He & Garner, 2025), which also augments GDN with a bounded evictable KV cache. The main difference is how the cache is selected and integrated: LTE learns eviction scores with an additional CNN module and alternates GDN with sparse-attention layers, whereas HOLA uses the delta rule's own write magnitude as a parameter-free surprise score and attaches the cache inside every recurrent layer. Thus LTE shows that evictable caches can help delta-rule models, while HOLA argues that the recurrent update itself already exposes the right signal for what an exact memory should keep.

Semiparametric and retrieval-augmented memory. kNN-LM (Khandelwal et al., 2020), RETRO (Borgeaud et al., 2022), Memorizing Transformers (Wu et al., 2022), and SPALM (Yogatama et al., 2021) pair a parametric LM with a non-parametric memory held in an external or long-lived datastore. HOLA is an in-sequence, test-time analogue: the recurrent state is the parametric part and a bounded set of exact in-context KV pairs the non-parametric part, between GDN (purely parametric) and full attention (unboundedly non-parametric).

3 Method

We first review the delta rule and why its state is lossy (Sec. 3.1), then present the HOLA layer (Sec. 3.2, Fig. 2), answer its two design questions — *what to store* (Sec. 3.3) and *how to read it* (Sec. 3.4) — and instantiate it on Gated DeltaNet (Sec. 3.5).

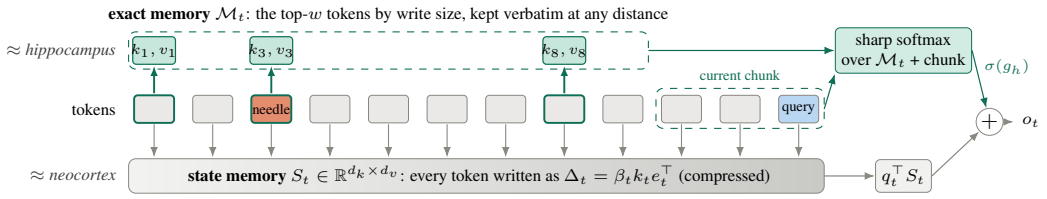


Figure 2: **HOLA along the token stream** (toy example, w=3). Every token is written into the recurrent state S_t as the delta-rule update $\Delta_t = \beta_t k_t e_t^\top$. The w tokens with the largest write so far (outlined; size $\beta \cdot \|e\|$) are also kept verbatim in the exact memory $\mathcal{M}_t$, however far back they are, e.g. the needle. The query reads the state ($q_t^\top S_t$) and, by a sharpened softmax, the exact memory and the current chunk; the two reads are added with a learned gate (Eq. 2).

3.1 Background: the delta rule and its lossiness

DeltaNet compresses history into a (d_k, d_v) matrix S, an *online associative memory*: the read-out for key k, $k^\top S$, gives its currently associated value. Let token t have query/key/value q_t, k_t, v_t (with q_t, k_t L2-normalized to unit norm) and write strength $\beta_t \in [0, 1]$. The delta rule updates by "predict, then write the residual":

$$S_t \;=\; S_{t-1} + \beta_t\, k_t\, e_t^\top, \qquad e_t \;=\; v_t - k_t^\top S_{t-1}, \qquad o_t^{\text{state}} \;=\; q_t^\top S_t. \tag{1}$$

Here e_t is the *residual* (innovation): the part of v_t that S cannot already predict along k_t. Writing the residual (rather than v_t) suppresses repeated same-key interference.

It is lossy. S is fixed-size, of rank at most d_k: once the number of distinct associations exceeds capacity, new writes *overwrite* old ones along shared key directions, and a distant (k, v) can no longer be recovered exactly from S. This is the "neocortex-like" deficiency of Sec. 1 — good at generalizing, unable to keep distant exact detail. The specific backbone we use (GDN) is deferred to Sec. 3.5.

3.2 HOLA: a state memory plus a bounded exact memory

HOLA gives every delta-rule layer two memories over the same keys and values (Fig. 2). The *state memory* is the backbone's recurrent state S_t (Eq. 1): every token is written into it, compressed. The *exact memory* $\mathcal{M}_t$ keeps verbatim (k_i, v_i) copies of at most w past tokens, selected by a per-token write score m_i that is fixed when token i is written. Its read also covers a chunk-local causal window — the earlier tokens of the current processing chunk of C tokens, so between 0 and $C-1$ of them — and one null sink entry; we denote this visible set, of at most $w+C+1$ entries, by $\mathcal{V}_t$. The chunk-local window is shared by the HOLA+recency control of Sec. 4, which differs only in how $\mathcal{M}_t$ is selected, so comparisons against it isolate the selection rule. The two reads are added with a learned per-head gate $\sigma(g_h)$, initialized small (gate init -4) so that the exact memory starts as a perturbation of the backbone:

$$o_t \;=\; o_t^{\text{state}} + \sigma(g_h)\, o_t^{\text{cache}}, \tag{2}$$

where o_t^{cache} is a softmax read over $\mathcal{V}_t$. The layer is thus determined by two choices: the write score m_i, i.e. *which pairs to keep* (Sec. 3.3), and *how sharply to read them* (Sec. 3.4).

3.3 What to store: the write magnitude $\beta \cdot \|e\|$ is "surprise"

We should store the tokens that are least well represented by the state memory. The *same update equation already says which*. Write the update as

$$S_t = S_{t-1} + \Delta_t, \qquad \Delta_t = \beta_t\, k_t e_t^\top \quad \text{(rank-1)}. \tag{3}$$

The token's entire effect on S is this one rank-1 matrix Δ_t. How much a token "writes" is naturally the size of Δ_t; for a rank-1 matrix the Frobenius norm factorizes, and with $\|k_t\| = 1$ collapses to a scalar:

$$m_t \;=\; \|\Delta_t\|_F \;=\; \beta_t\, \|k_t\|\, \|e_t\| \;=\; \beta_t\, \|e_t\|. \tag{4}$$

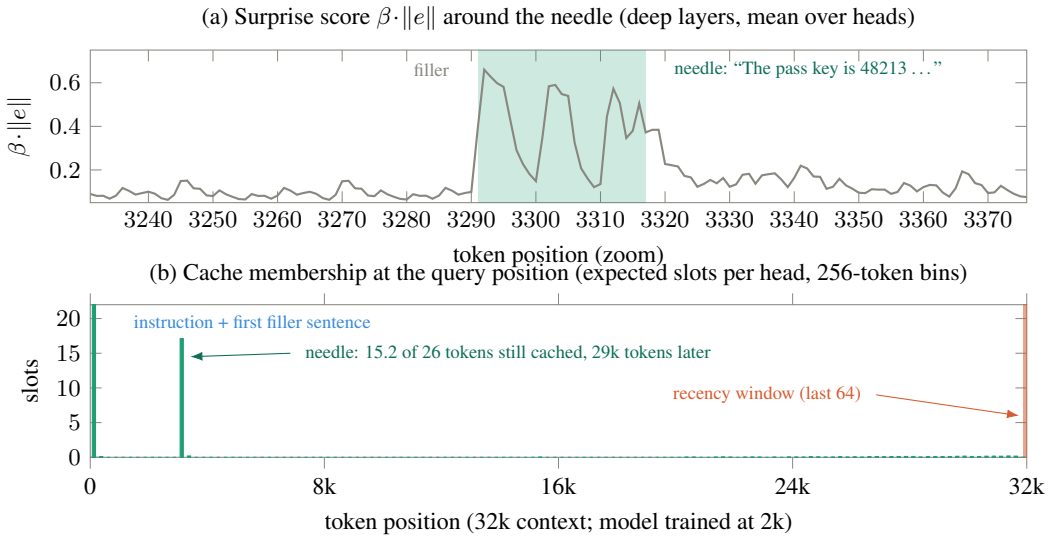


Figure 3: **Anatomy of the cache on one 32k NIAH sequence (trained 1B HOLA). (a)** The delta-rule write magnitude $\beta \cdot \|e\|$ (deep layers, mean over heads) around the needle: flat on filler, spiking on the needle tokens. **(b)** Expected number of cache slots per head occupied by each 256-token bin at the query position. The needle bin (position ≈3.3k) is still heavily cached at 32k; a recency window covers only the last 64 tokens.

We write this as $\beta \cdot \|e\|$ and use it directly as the eviction score. Its meaning is immediate: m_t **is how much the token changed** S**.** A large change means the token brought information S could *not* predict (large residual e_t, i.e. the innovation in the sense of Kalman (1960); the delta rule is the Widrow–Hoff/LMS rule (Widrow & Hoff, 1960), where e is its error term) and that the model actually wrote it in (large β_t). A token with small residual is already well predicted by the state, so storing a verbatim copy is wasteful; a token with large committed residual is exactly where the compressed state needed the most help. *In short, we spend the limited exact memory where the state-memory representation is weakest.*

Cache = the top-w tokens seen so far by $\beta \cdot \|e\|$. $\mathcal{M}_t$ holds the exact (k, v) copies of the w highest-m tokens observed before the current chunk (default w=64), *regardless of distance*. Because m_t is fixed when the token is written, this top-w set is order-independent, so training and inference share the same cache semantics.

Figure 3 shows what this rule does in the trained 1B model on a real 32k needle-in-a-haystack sequence. The score $\beta \cdot \|e\|$ is flat on the repeated filler and spikes on the needle tokens (panel a); at the query, 15.2 of the 26 needle tokens are still in the 64-slot cache 29k tokens later, and the remaining slots hold the instruction and the *first* occurrence of the filler sentence (panel b, Fig. 1a). In other words, $\beta \cdot \|e\|$ acts as a novelty detector: it keeps first occurrences regardless of distance, and repeated content — already predicted by the state — is never admitted. The picture is the same at 340M (15.5 of 26 needle tokens cached) and at 8k context.

Versus recency. Sliding-window memories, as used in StreamingLLM (Xiao et al., 2024), NHA (Du et al., 2026), and RAttention (Wang et al., 2025), choose exact KV pairs by position. This is useful for local context, but it cannot keep an old item solely because it remains important. Selection by $\beta \cdot \|e\|$ instead keeps surprising KV pairs across distance, compensating the state-memory failure described in Sec. 3.1. An ablation (Sec. 4.4, Table 3) shows why the product matters: the residual alone ($\|e\|$) or the write strength times the value norm ($\beta\|v\|$) each underperform; their product is best.

3.4 How to read: retrieval, not soft averaging

An exact copy is worthless if *read* like a linear attention. If the cache reuses the backbone's unit-L2-normalized q, k (Yang et al., 2024b; 2025), the effective logit is $\tau \cos / \sqrt{d}$ for head dimension d and temperature τ. For example, with $d{=}256$ and $\tau{=}6.6$ it is $\approx 0.41 \cos$, ranging only over ± 0.41: the softmax is nearly uniform — among $w{=}64$ entries a perfectly matching key receives only $\sim 2.3\%$ of the mass. The cache degenerates into yet another soft-averaging lossy summary (explaining why a naive cache barely helps).

Sharpening via a decoupled RMSNorm-γ. We apply Qwen3-style RMSNorm (Qwen Team, 2025) with a learnable γ to the *cache-path* q, k, keeping their norm near $\sqrt{d}$ rather than 1, and fix $\tau{=}1$:

$$o_t^{\text{cache}} = \sum_{j \in \mathcal{V}_t} \text{softmax}_j\Big(\tilde{q}_t^\top \tilde{k}_j / \sqrt{d}\Big)\, v_j, \qquad \tilde{q} = \text{RMSNorm}_\gamma(q), \;\; \tilde{k} = \text{RMSNorm}_\gamma(k). \tag{5}$$

The effective logit becomes $\approx \sqrt{d} \cos$ ($16 \cos$ for $d{=}256$), so the cache performs *near-argmax retrieval*. The change is decoupled from the state update, whose q, k stay unit-L2: the delta rule needs $\|k\| = 1$ to keep the eigenvalues of $I - \beta\, kk^\top$ in $[0, 1]$, and a $\sqrt{d}$ norm would give $1 - \beta d$ and diverge. **Sharpening is the largest lever for capacity in the design** ($\sim 2\times$ multi-key capacity, Table 8): on the 32k sequence of Fig. 3 it places 0.16 of the cache mass on the needle slots (up to 0.92 in single heads), versus 0.04 with unit-L2 q, k.

3.5 Instantiation

Instantiation (GDN). The derivation used only the delta rule. We instantiate it on Gated DeltaNet (GDN) (Yang et al., 2025), a widely used delta-rule backbone, which adds a data-dependent decay gate $\alpha_t \in (0, 1]$ (selective forgetting), making the prediction $\alpha_t k^\top S_{t-1}$ and the residual $e_t = v_t - \alpha_t k^\top S_{t-1}$. The gate is *orthogonal* to our method and $\beta \cdot \|e\|$ is unchanged; all experiments use GDN.

Overhead over GDN. The only new parameters are the cache-path RMSNorm scales and a per-head sink and gate: 12.6K parameters in our 1B model, about 0.001% of its 1.07B. The cache itself is inference state of at most $w{+}C$ KV pairs per layer, independent of context length: ~ 55 MB in bf16 at 1B, under 3% of the model's bf16 weights. Measured peak decode memory at 1B is 2.14 GB versus 2.07 GB for GDN ($+3.3\%$), flat from 2k to 128k context (Sec. 4.5, App. E).

4 Experiments

4.1 Setup

We train HOLA at two scales, 1B parameters on 50B tokens and 340M parameters on 15B tokens, both with a 2048-token training context. The two scales play complementary roles. **1B is our main scale**: it tests whether HOLA is competitive with the strongest published linear-attention models at a realistic size. **340M is our controlled scale**: there we train GDN, HOLA, and a HOLA+recency control on an identical backbone and recipe, so that their differences are attributable to the cache alone.

Both scales use the GDN architecture (Yang et al., 2025) with the same HOLA cache in every layer (cache size $w{=}64$, chunk $C{=}256$); architecture details, baselines, and the evaluation protocol are in App. A.

Training recipe. Both scales train on SlimPajama (Soboleva et al., 2023) with the Mistral tokenizer (Jiang et al., 2023) in a single pass, using AdamW (Loshchilov & Hutter, 2019) with peak lr $4{\times}10^{-4}$, $\beta{=}(0.9, 0.95)$, wd 0.01, grad-clip 1.0, a cosine schedule, and batches of 0.5M tokens (256 sequences of 2048 tokens). At 1B we train for 50.0B tokens (95,368 steps) with 1024 warmup steps and cosine decay to 10% of the peak lr, on $8{\times}$H200 with BF16 autocast over FP32 master weights. At 340M we train for 15.0B tokens (28,610 steps) with 1000 warmup steps on $8{\times}$A800, following the Preconditioned-DeltaNet 340M recipe (Tumma et al., 2026); this recipe match enables reuse of the published DeltaNet/KDA/GDN rows. The published 1B baselines share our 50B-token budget, but they are performance references rather than same-backbone controls.

Table 1: Main results. Published rows: Transformer++, GLA, GSA from Du et al. (2026); DeltaNet, GDN, KDA, PGDN, PKDA from Tumma et al. (2026). Retrieval Avg. is the four-task mean, Commonsense Avg. the six-task mean. Bold: best sub-quadratic result per scale. Protocol details: App. D.

	Retrieval ↑					LM ppl ↓		Commonsense ↑						
Model	FDA	SWDE	SQuAD	DROP	Avg.	Wiki	LMB	ARCe	ARCc	Hella	PIQA	Wino	LMBa	Avg.
1B scale / 50B tokens (Wiki = WikiText-2 word ppl)														
GDN	19.69	49.86	37.70	20.50	31.94	18.05	13.46	56.10	27.30	48.91	70.08	54.22	46.42	50.51
PGDN	18.33	50.32	**41.19**	20.80	32.66	18.01	13.09	56.66	26.96	**49.66**	70.73	54.54	46.49	50.84
KDA	23.23	50.14	37.00	19.70	32.52	17.95	14.52	54.55	26.88	48.95	70.29	54.22	45.16	50.01
PKDA	21.69	48.79	38.61	20.70	32.45	17.82	**11.86**	**57.24**	26.96	49.05	70.62	55.49	**48.52**	**51.31**
HOLA (ours)	**23.96**	**54.19**	40.78	**22.76**	**35.42**	**17.66**	12.88	55.89	**27.56**	48.82	**71.98**	**55.88**	47.56	51.28
340M scale / 15B tokens (Wiki = WikiText-103 token ppl)														
Transformer++	46.14	25.87	33.22	20.03	31.32	26.88	42.15	44.91	25.94	34.95	64.31	51.07	32.84	42.34
DeltaNet	8.53	27.09	28.32	17.80	20.44	29.04	45.76	44.02	23.55	34.08	65.07	50.91	29.58	41.20
GLA	11.26	16.78	27.85	17.68	18.39	28.78	39.00	44.53	22.27	34.84	63.93	51.38	32.27	41.54
GSA	6.36	16.87	21.90	16.72	15.46	28.17	42.57	45.50	24.23	35.00	64.85	50.43	30.78	41.80
KDA	13.88	34.11	29.69	16.90	23.65	26.18	31.37	45.45	22.70	**36.06**	**66.00**	52.25	34.04	42.75
GDN	13.61	29.43	29.83	17.20	22.52	27.08	31.39	44.49	24.32	35.96	65.83	51.30	34.50	42.73
GDN (ours)	11.70	29.00	32.50	18.10	22.83	27.32	30.95	**46.13**	23.72	35.88	65.07	50.43	34.02	42.54
HOLA+recency	16.90	29.90	30.60	17.30	23.68	25.04	32.33	**46.13**	**24.40**	35.62	65.34	**52.80**	**34.72**	**43.17**
HOLA (ours)	**20.20**	**35.90**	**34.10**	**20.80**	**27.75**	**22.92**	**30.27**	46.00	24.06	35.91	65.02	51.54	34.54	42.85

4.2 Main results: retrieval, language modeling, commonsense

Table 1 reports both scales. **At 1B/50B, HOLA is the strongest linear model on in-context retrieval by a clear margin.** It scores 23.96/54.19/40.78/22.76 on FDA/SWDE/SQuAD/DROP, exceeding every published baseline on three of the four tasks (PGDN leads SQuAD, 41.19 vs. 40.78) and lifting the four-task average from at most 32.66 (PGDN) to 35.42 (on these tasks our evaluation reproduces the published 340M GDN average to within 0.3 points; App. D); its Wikitext perplexity (17.66) is the lowest in the block and its six-task commonsense mean (51.28) matches the best baseline (PKDA, 51.31). The 340M block provides the controlled comparison: GDN, HOLA+recency, and HOLA share an identical backbone and recipe and differ only in the cache. The picture is the same there: **HOLA substantially boosts in-context retrieval** (four-task average 22.83 → 27.75), Wikitext perplexity drops 27.32 → 22.92 under the standard protocol and 14.51 → 14.20 without paragraph BOS tokens (App. G explains the difference), and commonsense stays competitive.

Three observations on the controlled 340M block:

- **Retrieval:** on in-context exact extraction HOLA beats *all linear* baselines by a wide margin — four-task average 27.75 vs. 23.68 for the HOLA+recency control and at most 23.65 for published linear models (KDA); FDA 11.7 → 20.2 (+73% rel.), SWDE 29.0 → 35.9 (+24%), SQuAD 32.5 → 34.1, DROP 18.1 → 20.8 over the GDN anchor. Only the full-attention Transformer++ (FDA 46.1) still leads pure extraction.
- **Perplexity:** HOLA has the lowest Wiki (22.92 vs. 27.32 for GDN) and LAMBADA (30.27) perplexity in the block, but most of the Wiki gain comes from paragraph-boundary BOS tokens (App. G); without them it is 2% (14.51 → 14.20; App. C).
- **Commonsense:** the six-task averages of HOLA (42.85), HOLA+recency (43.17), KDA (42.75), and GDN (∼42.7) lie within 0.6 points; commonsense does not separate the models.

4.3 Long-Context Retrieval

We evaluate on the official RULER benchmark (Hsieh et al., 2024) (synthetic tasks, our Mistral tokenizer, limit 100) at 2k/4k/8k/16k/32k — up to 16× the 2k training length. On the headline single needle-in-a-haystack task, S-NIAH-1, **the recurrent state (GDN) collapses with length while HOLA stays robust**, with a matched HOLA+recency cache in between (Figure 1b): at 32k, HOLA reaches 0.58 vs. HOLA+recency 0.24 vs. GDN 0.14 — a +0.44 margin over the state, and +0.64 at 16k.

The advantage extends to most above-floor RULER cells (Table 2), grouped into single-needle (S-NIAH-1/2/3) and multi-needle (multi-key/-value/-query): HOLA wins the harder single-needle variants by wide margins where they are not floored (S-NIAH-2 at 8k, 0.35 vs. 0.09) and most multi-needle cells (multi-value 0.28 vs. 0.17 at 2k), with a few near-tie exceptions. We also include a full-attention Transformer++ as a 2k full-attention ceiling: within its training length it is the strongest model (exact softmax retrieval), but this RoPE checkpoint is not a length-extrapolating baseline and every shown task drops to 0 at 4k and beyond — exactly the regime where HOLA remains useful.

Table 2: RULER accuracy at 340M, 2k–8k. Single-needle: S-NIAH-1/2/3; multi-needle: multi-key (MK1), multi-value (MV), multi-query (MQ). Bold: best sub-quadratic model per cell. Transformer++ (2k-trained, RoPE) is a full-attention reference; S-NIAH-1 at 16k/32k is in Fig. 1b.

		Single-needle			Multi-needle		
len	model	1	2	3	MK1	MV	MQ
2k	Transformer++	1.00	1.00	0.79	0.71	0.34	0.33
	GDN	1.00	0.38	0.26	0.17	0.17	**0.24**
	HOLA+recency	1.00	0.83	0.89	**0.26**	0.21	0.22
	HOLA	1.00	**1.00**	**0.96**	0.25	**0.28**	0.18
4k	Transformer++	0.00	0.00	0.00	0.00	0.00	0.00
	GDN	**1.00**	0.52	0.23	0.17	0.14	0.19
	HOLA+recency	0.93	0.27	0.38	0.13	0.17	0.17
	HOLA	0.99	**0.89**	**0.43**	**0.30**	**0.28**	**0.26**
8k	Transformer++	0.00	0.00	0.00	0.00	0.00	0.00
	GDN	0.83	0.09	**0.07**	0.03	0.07	0.02
	HOLA+recency	0.74	0.04	0.02	0.05	0.03	0.01
	HOLA	**0.98**	**0.35**	0.05	**0.11**	**0.16**	**0.08**

4.4 Controls and ablations

HOLA+recency vs. surprise-based eviction (340M, matched memory). The sharpest test of the eviction rule is a matched control — the identical architecture and cost (w=64, the same chunk-local window, sharpened read, gate, and kernel), changing *only* the eviction signal: *position* (keep the most recent w) vs. *surprise* (keep top-w by $\beta \cdot \|e\|$). We report the HOLA+recency control as a first-class row throughout, and surprise-based eviction wins on the axes the cache is designed to affect: in-context retrieval (four-task average $23.68 \rightarrow 27.75$), perplexity under both protocols (Wiki $25.04 \rightarrow 22.92$ standard, $14.36 \rightarrow 14.20$ without paragraph BOS; Table 1 and App. C), and long-context retrieval (Figure 1b, Table 2). Most tellingly, on the far needle the recency cache is barely better than no cache at all (Sec. 4.3): the needle slides out of the recency window, whereas surprise-based eviction keeps it. For a bounded exact memory, *what* to cache matters more than *how recent*.[1]

Eviction signal (Table 3). We isolate the eviction rule with the same 46M backbone, cache size, and flat cache read, and evaluate the two axes a bounded cache is meant to balance: whether the state still carries a far needle after the cache window has moved on, and how many exact key–value facts the cache can retain within its span. The far-needle column uses a lightweight passkey-style probe inspired by Mohtashami & Jaggi (2023): we place a numeric key at a controlled depth in a 4k context, teacher-force the answer tokens after the query, and report next-token accuracy. Pure residual surprise, $\|e\|$, is not enough; it lacks the write-strength utility signal and performs poorly on the far needle. Multiplying by the write strength β gives $\beta \cdot \|e\|$, the actual delta-rule update magnitude, which is best or tied-best on every diagnostic column while also giving the lowest standard-protocol WikiText perplexity among the cache variants. With paragraph BOS tokens removed, no flat-read cache helps at 46M: all are worse than the GDN anchor (38.31), and all but recency lie above GDN's three-seed range 37.7–40.2, so at 46M the eviction rule is decided by the recall diagnostics, not by perplexity.

[1]For the recency control, we use full recomputation to avoid implementation-path confounds; this is the path used for the PPL, commonsense, retrieval, and RULER numbers we report.

Table 3: Eviction-signal diagnostic (46M, flat read). Far is teacher-forced passkey accuracy at depth 0.1 in a 4k context; multi-key columns report associative-recall capacity for 1/2/4 facts; perplexity is WikiText-103 test under the standard protocol.

Eviction score	has β?	far d=0.1	1 key	2 keys	4 keys	Wiki PPL ↓
GDN (no cache)	–	0.55	0.78	0.45	0.41	70.21
cumulative attention (H2O)	–	0.43	**0.97**	0.69	0.55	71.45
residual $\|e\|$	no	0.22	0.71	0.61	0.48	70.50
$\beta\|v\|$	yes	0.42	**0.97**	0.72	0.54	70.76
$\beta \cdot \|e\|$ **(ours)**	yes	**0.67**	**0.97**	**0.74**	**0.56**	**70.10**

Sharpened read. With the eviction rule fixed to $\beta \cdot \|e\|$, replacing the unit-L2 cache read by the decoupled RMSNorm-γ read roughly doubles multi-key capacity (4 keys $0.56 \to 0.77$, 8 keys $0.41 \to 0.67$) without losing the far-needle behavior of the recurrent state; the full ablation is in App. F.

4.5 Efficiency

Table 4 summarizes the cost side at our main scale (1B, one H200, bf16; GDN and Transformer++ are same-size randomly initialized models, as timing and memory do not depend on weights). With fused cache kernels (App. E), a HOLA training step costs $1.19\times$ a GDN step and prefill at 32k $1.24\times$ (214 vs. 173 ms; $\sim 1.7\times$ at short lengths, where kernel launches dominate). Decode memory stays within 3.3% of GDN and flat to 128k, while full attention's grows to 24.5 GB and its decode throughput falls behind both linear models once batch or context grows (Fig. 5). The retrieval gains are not bought with this cost: the HOLA+recency control has the same overhead yet stays far behind (Sec. 4.4), and the published GDN gains 9.4 retrieval points from the 340M/15B to the 1B/50B setting, roughly an order of magnitude more training compute, whereas HOLA adds 4.9 at 340M for $1.19\times$ the training cost.

Table 4: Efficiency summary (1B, one H200, bf16; HOLA decode uses an exact fused eager kernel, no CUDA Graph; training and prefill use the fused implementation of App. E). Prefill is the median of eleven runs; decode numbers are the best of three runs. Peak memory is measured during bs=1 decoding with weights included.

Metric	Full attention	GDN	HOLA
Training step, 16×2k (rel. GDN)	**0.77**×	1.0×	1.19×
Prefill latency, bs1 @ 32k (ms) ↓	373	**173**	214
Decode latency, bs1 @ 32k (ms/tok) ↓	**15.7**	22.3	30.1
Decode throughput, bs8 @ 32k (tok/s) ↑	159	**352**	264
Decode throughput, bs32 @ 32k (tok/s) ↑	OOM	**1410**	1024
Peak memory, bs1 @ 32k / 128k (GB) ↓	8.11 / 24.5	**2.07 / 2.07**	2.14 / 2.14

5 Conclusion

Following Complementary Learning Systems theory, we attached a bounded exact KV cache, a hippocampus-like store beside the neocortex-like recurrent state, to every layer of Gated DeltaNet. The cache is guided by two design choices: (i) eviction driven by a *parameter-free, intrinsic* surprise signal $\beta \cdot \|e\|$ (the delta-rule write magnitude) — keep what wrote most to the state, not what is most recent; and (ii) a read sharpened by a *decoupled* RMSNorm-γ, so the cache performs exact retrieval rather than soft averaging. At 1B parameters trained on 50B tokens, HOLA surpasses every published linear baseline we compare on in-context retrieval (four-task average 35.4 vs. at most 32.7) with the lowest Wikitext perplexity and matching commonsense; at 340M, against matched controls, it retains far more RULER single-needle recall out to 32k than GDN or a recency cache — at 3.3% more peak decode memory than GDN at 1B, flat to 128k. More broadly, a delta-rule model's own update already identifies the tokens its state fails to absorb, and a small exact memory spent on them recovers much of the exact recall that linear attention loses.

AI USE STATEMENT

In this work, we used generative AI tools for all tasks that require disclosure: to propose and refine hypotheses, help develop the conceptual framework, design and provide feedback on the research methodology and experiments, implement methods, generate synthetic datasets, clean and reformat datasets, assist with translation, formulate mathematical claims, provide ingredients for proving mathematical claims and assist in writing the derivations, support qualitative and thematic data analysis, and interpret results; all interpretations reported in the paper were reviewed by the authors. Additionally, we used generative AI tools to suggest experimental parameters, create and edit software code, create and modify scientific figures, suggest the structure of the paper, draft and edit parts of the paper, and search for, identify, and summarize related literature. We have reviewed all AI-assisted work: AI-assisted code was tested for correctness, and citations and factual claims were checked against the source material. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The HOLA layer is specified in Section 3 and the training recipe in Section 4; architecture configurations, baselines, and evaluation protocols are in App. A and App. D, and the fused implementation used for the efficiency measurements, together with its validation, is in App. E. We will release code, model configurations, evaluation scripts, and checkpoints upon publication.

REFERENCES

Simran Arora, Sabri Eyuboglu, et al. Simple linear attention language models balance the recall-throughput tradeoff. In *ICML*, 2024a. arXiv:2402.18668.

Simran Arora, Sabri Eyuboglu, et al. Zoology: Measuring and improving recall in efficient language models. In *ICLR*, 2024b. arXiv:2312.04927.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. In *NeurIPS*, 2025. arXiv:2501.00663.

Sebastian Borgeaud et al. Improving language models by retrieving from trillions of tokens. In *ICML*, 2022. arXiv:2112.04426.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *ICML*, 2024. arXiv:2405.21060.

Soham De et al. Griffin: Mixing gated linear recurrences with local attention for efficient language models. *arXiv:2402.19427*, 2024.

Jusen Du, Jiaxi Hu, Tao Zhang, Weigao Sun, and Yu Cheng. Native hybrid attention for efficient sequence modeling. In *ACL*, 2026. arXiv:2510.07019.

Yunhao Fang, Weihao Yu, Shu Zhong, Qinghao Ye, Xuehan Xiong, and Lai Wei. Artificial hippocampus networks for efficient long-context modeling. *arXiv:2510.07318*, 2025.

Paul W. Frankland and Bruno Bontempi. The organization of recent and remote memories. *Nature Reviews Neuroscience*, 6(2):119–130, 2005.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *COLM*, 2024. arXiv:2312.00752.

Mutian He and Philip N. Garner. Alleviating forgetfulness of linear attention by hybrid sparse attention and contextualized learnable token eviction. *arXiv:2510.20787*, 2025.

John J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *PNAS*, 79(8):2554–2558, 1982.

Cheng-Ping Hsieh et al. RULER: What's the real context size of your long-context language models? In *COLM*, 2024. arXiv:2404.06654.

Samy Jelassi, David Brandfonbrener, Sham M. Kakade, and Eran Malach. Repeat after me: Transformers are better than state space models at copying. In *ICML*, 2024. arXiv:2402.01032.

Albert Q. Jiang et al. Mistral 7B. *arXiv:2310.06825*, 2023.

Rudolph E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *ICML*, 2020. arXiv:2006.16236.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. In *ICLR*, 2020. arXiv:1911.00172.

Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, et al. Kimi Linear: An expressive, efficient attention architecture. *arXiv:2510.26692*, 2025.

Dharshan Kumaran, Demis Hassabis, and James L. McClelland. What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in Cognitive Sciences*, 20(7):512–534, 2016.

Opher Lieber et al. Jamba: A hybrid Transformer-Mamba language model. *arXiv:2403.19887*, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. arXiv:1711.05101.

James L. McClelland, Bruce L. McNaughton, and Randall C. O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological Review*, 102(3):419–457, 1995.

Amirkeivan Mohtashami and Martin Jaggi. Landmark attention: Random-access infinite context length for transformers. In *NeurIPS*, 2023. arXiv:2305.16300.

Qwen Team. Qwen3 technical report. *arXiv:2505.09388*, 2025.

Hubert Ramsauer et al. Hopfield networks is all you need. In *ICLR*, 2021. arXiv:2008.02217.

Liliang Ren et al. Samba: Simple hybrid state space models for efficient unlimited context language modeling. In *ICLR*, 2025. arXiv:2406.07522.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *ICML*, 2021. arXiv:2102.11174.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R. Steeves, Joel Hestness, and Nolan Dey. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama. https://www.cerebras.ai/blog/slimpajama-a-627b-token-cleaned-and-deduplicated-version-of-redpajama, 2023.

Dorothy Tse, Rosamund F. Langston, Masaki Kakeyama, Ingrid Bethus, Patrick A. Spooner, Emma R. Wood, Menno P. Witter, and Richard G. M. Morris. Schemas and memory consolidation. *Science*, 316(5821):76–82, 2007.

Neehal Tumma, Noel Loo, and Daniela Rus. Preconditioned DeltaNet: Curvature-aware sequence modeling for linear recurrences. *arXiv:2604.21100*, 2026.

Bailin Wang, Chang Lan, Chong Wang, and Ruoming Pang. RATTENTION: Towards the minimal sliding window size in local-global attention models. *arXiv:2506.15545*, 2025.

Bernard Widrow and Marcian E. Hoff. Adaptive switching circuits. In *IRE WESCON Convention Record*, 1960.

Yuhuai Wu, Markus N. Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers. In *ICLR*, 2022. arXiv:2203.08913.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *ICLR*, 2024. arXiv:2309.17453.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In *ICML*, 2024a. arXiv:2312.06635.

Songlin Yang, Bailin Wang, Yu Zhang, et al. Parallelizing linear transformers with the delta rule over sequence length. In *NeurIPS*, 2024b. arXiv:2406.06484.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. In *ICLR*, 2025. arXiv:2412.06464.

Dani Yogatama, Cyprien de Masson d'Autume, and Lingpeng Kong. Adaptive semiparametric language models. *Transactions of the ACL (TACL)*, 2021. arXiv:2102.02557.

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 *NeurIPS*, 2024. arXiv:2409.07146.

Zhenyu Zhang et al. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *NeurIPS*, 2023. arXiv:2306.14048.

A Experimental setup details

Architecture. Both scales use the GDN architecture (Yang et al., 2025) with head dimension 256, expand_v=1, hidden_ratio=4, short convolution of size 4, a Swish output gate, tied embeddings, and vocabulary 32000. The 1B model has 1.071B parameters: d_{model}=1792, 24 layers, and 7 heads. The 340M model follows the GDN 340M configuration: d_{model}=1024, 24 layers, and 4 heads. At both scales HOLA adds the same cache to every layer: cache size w=64, chunk C=256, eviction score $\beta \cdot \|e\|$, cache normalization RMSNorm-γ, τ=1 frozen, and gate init -4. At 340M the GDN baseline and the HOLA+recency control *share an identical backbone* with HOLA; the only difference is the cache (or, for the control, its eviction rule).

Baselines. At 1B we compare against the published 1B/50B rows of Tumma et al. (2026) (GDN, KDA, and their preconditioned variants PGDN and PKDA; KDA itself is from Kimi Linear (Kimi Team et al., 2025)). At 340M we train our own GDN anchor and HOLA for the controlled same-backbone comparison, and borrow recipe-matched rows for Transformer++, GLA, and GSA from Du et al. (2026), and DeltaNet, KDA, and GDN from Tumma et al. (2026).

Evaluation. (1) In-context retrieval: FDA, SWDE, SQuAD, and DROP. (2) Language modeling: Wikitext perplexity under each block's published protocol — WikiText-2 word perplexity (lm-eval) at 1B, and WikiText-103 token perplexity under the standard fla protocol at 340M (each paragraph tokenized separately, so a BOS token precedes every paragraph) — with BOS-free perplexities in App. C, and LAMBADA. (3) Commonsense reasoning (zero-shot): ARC-e/c, HellaSwag, PIQA, WinoGrande, and LAMBADA accuracy. (4) Long context, on the controlled 340M models: RULER (Hsieh et al., 2024) at 2k→32k, with single-needle S-NIAH-1/2/3 and multi-needle multi-key (MK1), multi-value (MV), and multi-query (MQ) tasks.

B Scale configurations and perplexity across scale

Table 5 reports the perplexity comparison across 46M, 170M, and 340M; each row uses a matched GDN-vs-HOLA backbone. Under the standard protocol HOLA lowers Wikitext perplexity by 15–16% at every scale; without paragraph BOS tokens the gain is small (-1.0% at 170M, -2.1% at 340M) and within seed noise at 46M (three-seed means 38.7 vs. 39.0), where the single-seed flat-read cache variants are slightly worse than GDN. The standard-protocol gain is discussed in App. G.

Table 6 lists the architecture, corpus size, and context length for the scaling comparison in Table 5. Within each row, GDN and HOLA use the same backbone and training recipe; HOLA only adds the bounded exact KV cache.

Table 5: HOLA vs. same-backbone GDN, Wikitext perplexity across scales under the standard protocol (46M rows are three-seed means).

Scale	GDN ppl ↓	HOLA ppl ↓
46M	71.0	**59.5**
170M	35.98	**30.51**
340M	27.32	**22.92**

Table 6: Scale configurations for Table 5.

Scale	d_{model}	layers	corpus	train tokens	ctx
46M	512	8	FineWeb-Edu	0.5B	4096
170M	1024	12	SlimPajama	6.22B	2048
340M	1024	24	SlimPajama	15.0B	2048

For 170M and 340M, the architecture family follows the GDN recipe: 4 heads $\times$ head-dim 256, $\text{expand_v}{=}1$, $\text{hidden_ratio}{=}4$, conv 4, tied embeddings, vocabulary 32000, Mistral tokenizer, and AdamW with peak lr $4{\times}10^{-4}$. The 46M row uses a smaller 8-layer, $d_{\text{model}}{=}512$ architecture (4 heads $\times$ head-dim 128) trained on FineWeb-Edu for 0.5B tokens at ctx 4096; this is the scale used for the component studies in Tables 3 and 8.

HOLA cache hyperparameters: eviction score $\beta \cdot \|e\|$, cache size $w{=}64$, chunk $C{=}256$, gate init -4, RMSNorm-γ cache normalization, and $\tau{=}1$ (frozen). The GDN baseline shares all other settings within each row.

C WIKITEXT PERPLEXITY WITHOUT PARAGRAPH BOS TOKENS

Appendix G shows that the standard fla Wikitext protocol inserts a BOS token before every paragraph and that HOLA's exact memory preserves context across the resulting resets of the recurrent state. This appendix collects the same checkpoints evaluated on the same test text without the inserted BOS: "rows, no BOS" tokenizes each row separately with no special tokens and concatenates (the only change from the standard protocol), and "single stream" tokenizes the whole test set as one text. All values are WikiText-103 test perplexity on 2048-token blocks.

Table 7: Wikitext-103 test perplexity of the 340M models under three tokenizations.

Model	standard (BOS per row)	rows, no BOS	single stream
GDN	27.32	14.89	14.51
HOLA+recency	25.04	14.71	14.36
HOLA	**22.92**	**14.51**	**14.20**

D EVALUATION PROTOCOL DETAILS FOR TABLE 1

Our 340M Wiki scores use WikiText-103 subword-token perplexity, whereas our 1B Wiki score uses WikiText-2 word perplexity; published Wiki values are retained as reported. These are not a cross-scale perplexity comparison, and exact equivalence to published evaluators is not established. No same-backbone 1B GDN control was trained, so the 1B rows do not isolate the cache. Our 340M retrieval results use the first 1000 examples per task, with 48 generated tokens for FDA/SWDE/DROP and 256 for SQuAD. Our 1B results use the full FDA/SWDE/SQuAD/DROP splits (1102/1111/2984/2087 examples), greedy generation with a 48-token budget, and case-insensitive answer containment. Prompts are left-truncated to fit context 2048 after reserving generation space. DROP is an in-context, single-read task scored by answer containment, not standard DROP EM/F1. We omit TriviaQA: our single-read in-context protocol scores the same 340M GDN recipe far above the published evaluation (44.9 vs. 33.8), so TriviaQA values are not comparable across sources. Published sample limits and truncation details have not been fully matched. On the

four retrieval tasks we report, however, the residual protocol difference is small: our evaluation of the same 340M GDN recipe gives a four-task average of 22.83 versus the published 22.52 (FDA -1.9, SWDE -0.4, SQuAD $+2.7$, DROP $+0.9$), an offset of 0.3 points against HOLA's 2.8-point margin at 1B. ARCe, PIQA, Wino, and LMBa use accuracy; ARCc and Hella use normalized accuracy.

E Efficiency details

Setup. All measurements use one H200 in bf16 with the 1B configuration of Sec. 4: HOLA is the trained checkpoint, GDN is the same architecture without the cache (1.071B parameters), and Transformer++ has the same hidden size and depth with 14 heads of dimension 128 (0.993B parameters); the latter two are randomly initialized, which does not affect timing or memory. HOLA timings use the fused implementation described below. Each benchmark runs alone on the node. Training throughput is fwd+bwd+AdamW at a constant 32,768 tokens per step; prefill is the median of eleven runs; decode uses chunked prefill to build the context and reports the best of three runs of 64 generated tokens, because eager decode at these sizes is launch-bound and sensitive to host-side jitter.

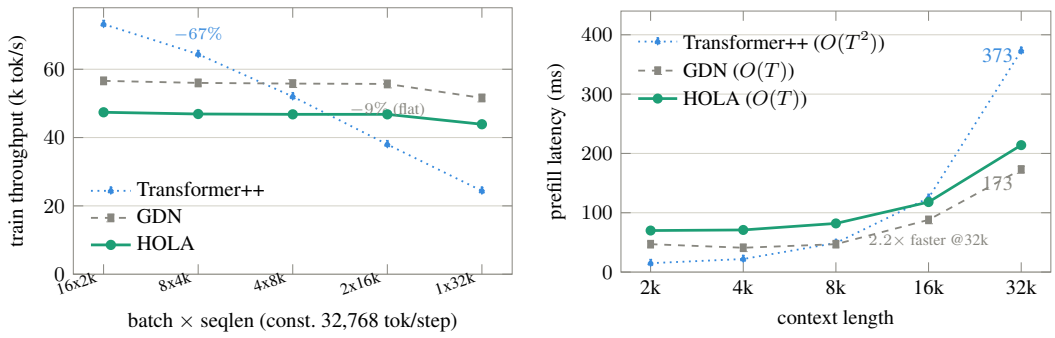
Reference implementation. Because the eviction score is static, the cache at every chunk is exactly the prefix top-w, so a per-chunk loop ($\approx 5\times$ GDN in our 340M development runs) collapses into one batched top-w selection and one batched attention over all chunks. Our reference implementation, used to train and evaluate every model in this paper, computes $\beta \cdot \|e\|$ with a second delta-rule pass and reads the cache with PyTorch SDPA over concatenated keys and an additive mask. At 1B it costs $1.58\times$ GDN per training step ($16\times$2k; the 340M main training run measured $1.50\times$ on $8\times$A800) and $1.93\times$ GDN in prefill at 32k. Profiling shows that the cache attention itself is a small part of this: of the 337 ms training overhead, $\sim$190 ms is concatenation, masking, and copies, and most of the 122 ms attention time is a slow masked backward; in prefill, the attention takes $\sim$16 of $\sim$160 ms.

Fused implementation. Three exact changes remove this overhead. (i) The score needs no second pass: the chunked delta-rule kernel already forms each token's write $\beta_t e_t$ (its per-token "new value"), so $\beta \cdot \|e\|$ is read off as its norm. (ii) The cache read runs as two FlashAttention varlen calls, chunk-local causal attention and attention to the chunk's top-w cache, whose log-sum-exps are merged with the sink in one fused kernel; the backward follows ring attention, differentiating each partial attention with the merged output and log-sum-exp, index-adding the cache-key gradients back to their token positions, and computing the sink gradient in closed form. (iii) Each chunk's top-w prefix set is found in two stages, per-chunk top-w and then top-w over earlier chunks' candidates, which selects the same set. The training step drops from 913 to 692 ms ($1.19\times$ GDN; peak memory $86 \rightarrow 72$ GB) and 32k prefill from 340 to 214 ms (Fig. 4, Table 4). The fused implementation reproduces the reference within bf16 noise. For inference, WikiText-103 perplexity is 16.410 versus 16.411, with 98.7% top-1 agreement, the same as the reference path's agreement with itself under a different batch composition (98.8%), because bf16 differences occasionally flip a borderline top-w member. For training, the per-operation gradient errors against an fp32 reference match those of the bf16 reference implementation; full-model gradients have cosine similarity 0.99968 to the reference, versus 0.99966 for the reference itself under a different batch split; and 26 steps of continued training from the 1B checkpoint give losses within 6×10^{-4} of the reference at every step. All reported accuracies come from models trained and evaluated with the reference implementation.

Decoding. Each step carries the GDN state plus a fixed-shape bounded-cache tuple (the committed top-w keys, values, and scores, the current chunk buffer, and a position). One eager Triton kernel fuses the surprise score, the cache-path RMSNorm-γ, the softmax reduction over the visible set, and the in-place cache update; it changes neither cache membership nor logits and matches full recomputation token for token. We exclude CUDA Graph capture from Table 4 because GDN and Transformer++ would need analogous graph-safe serving paths for a fair comparison.

F Read-side ablation: sharpened cache read

With the eviction rule fixed to $\beta \cdot \|e\|$ (Table 8), the decisive read-side change is the normalization used to form cache logits. Unit-L2 queries and keys make the cache read too flat, so the cache behaves like a soft average. RMSNorm-γ keeps the natural $\sqrt{d}$ logit scale while letting the model tune it, turning the same bounded cache into an exact local memory: it sharply improves multi-key ca-



(a) training throughput (b) prefill latency

Figure 4: **Training and prefill scaling (1B, one H200). (a)** Training throughput at a constant 32,768 tokens per step. **(b)** Prefill latency at bs=1. Full attention is cheapest up to ∼8k and 2.2× slower than GDN at 32k. With fused cache kernels, HOLA's training step is 1.19× GDN's and its prefill is 1.24× GDN's at 32k.

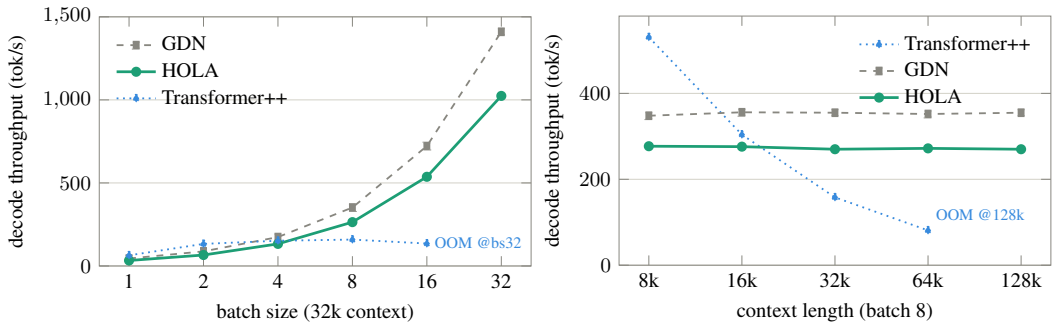


Figure 5: **Decode-serving crossover (1B, one H200; best of three runs).** *Left:* aggregate decode throughput vs. batch at 32k context. *Right:* throughput vs. context at batch 8. Full attention wins only in the launch-bound small-batch, short-context corner; with batching or long context, the bounded state and cache of GDN and HOLA overtake it, and full attention runs out of memory at batch 32 and at 128k context.

pacity without losing the far-needle behavior of the recurrent state. It also lowers standard-protocol WikiText perplexity by more than ten points, but without paragraph BOS tokens the RMSNorm-γ read (39.0, three-seed mean) is within GDN's seed range (37.7–40.2; seed-42 anchor 38.3) and the unit-L2 read is slightly worse (40.7), so the read-side gain is in capacity, not in language modeling.

Table 8: Cache-read normalization (46M). The GDN and unit-L2 rows repeat the flat-read anchors from Table 3 where columns overlap; the RMSNorm-γ row is a three-seed mean. Far is teacher-forced passkey accuracy at depth 0.1 in a 4k context; mean is the passkey average over depths; capacity reports multi-key associative recall. Wiki PPL is under the standard protocol.

Read / model	Wiki PPL ↓	far d=0.1	passkey mean	1 key	4 keys	8 keys	16 keys
GDN (no cache)	70.21	0.55	0.60	0.78	0.41	0.35	0.31
$\beta \cdot \|e\|$ + unit-L2 read	70.10	0.67	0.67	0.97	0.56	0.41	0.31
$\beta \cdot \|e\|$ **+ RMSNorm-γ (ours)**	**59.5**	**0.75**	**0.69**	**0.97**	**0.77**	**0.67**	**0.41**

G THE EXACT MEMORY ACROSS DOCUMENT BOUNDARIES

Under the standard fla perplexity protocol, HOLA improves Wikitext perplexity over its GDN backbone by 16% (Table 1); without the paragraph BOS tokens it is 2% (App. C). This appendix shows

that the difference is a direct demonstration of the exact memory's intended role. The standard script tokenizes each Wikitext row separately and concatenates the results, so a BOS token precedes every paragraph (about one token in 80); all published 340M rows in Table 1 share this protocol, so the comparison is fair, but the protocol makes every paragraph boundary look like a document boundary. GDN's data-dependent decay gate, trained on packed SlimPajama where BOS separates unrelated documents, has learned to reset the state there: its mean gate is $\alpha \approx 0.05$ on BOS tokens versus 0.80 elsewhere (Fig. 6b), so after every paragraph the recurrent state has discarded the article so far. HOLA's state resets in the same way, but its exact memory does not, and it carries the article's salient tokens — headings, names, first occurrences — across the boundary. Accordingly, HOLA's per-token gain over GDN is largest in the first tokens after a boundary and decays with distance (Fig. 6a); tokenizing the same rows without the inserted BOS moves GDN from 27.32 to 14.89 and HOLA from 22.92 to 14.51 (Table 7). The recurrent state forgets at boundaries by design; the hippocampal memory is exactly what remembers across them. Because the size of this effect depends on how often an evaluation inserts such boundaries, we base the paper's claims on retrieval and length robustness and report Wikitext perplexity under both protocols.

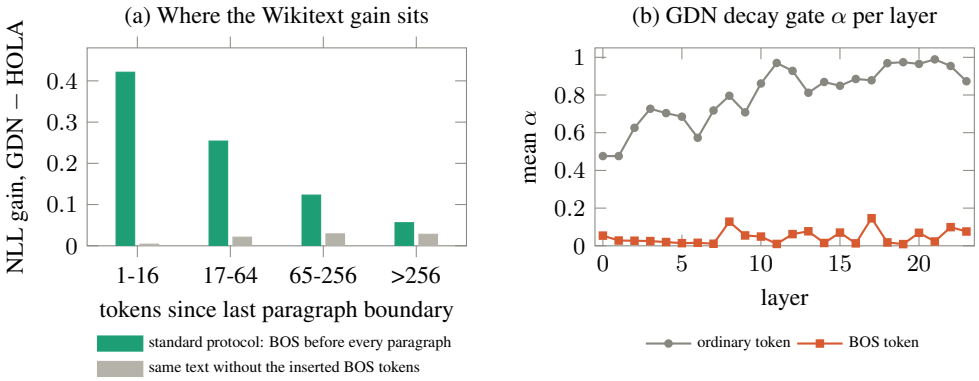


Figure 6: **The exact memory carries context across document-boundary resets.** **(a)** Per-token NLL gain of HOLA over GDN (340M) as a function of distance from the last paragraph boundary, with and without the BOS token the protocol inserts there. **(b)** GDN's mean decay gate α per layer on BOS tokens versus ordinary tokens: the recurrent state resets at every BOS, the exact memory does not.