BEYOND STATE CAPACITY: DIAGNOSING AND IMPROVING MULTI-SLOT LINEAR ATTENTION THROUGH SELECTIVITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-slot linear attention is commonly understood as expanding recurrent-state capacity, but this view alone cannot explain why routing designs with comparable state capacity differ in retrieval performance. We show that multi-slot routing is not merely state-capacity expansion: it factorizes a query-conditioned selection operator over historical tokens. Under this view, routing design determines the operator's usable expressivity even at fixed state capacity. We identify two structural restrictions in existing methods: cross-token selectivity leakage, where selecting one token also emphasizes others sharing its slot, and tied selectivity, where coupled read and write routing limits which selection patterns can be expressed. These diagnoses lead to **SimpleRoute**, a simple routing mechanism that removes both restrictions to improve selective retrieval while retaining linear complexity in sequence length. Trained from scratch with 1B parameters on 50B tokens, SimpleRoute achieves the highest average commonsense and real-world retrieval scores among the compared large-scale recurrent models—including 1.3B models trained on twice as many tokens—and improves real-world retrieval by 6.35 average points over the strongest baseline at the same scale and token budget.

1 INTRODUCTION

Linear attention (Katharopoulos et al., 2020; Choromanski et al., 2021; Yang et al., 2024a) compresses key–value history into fixed-size recurrent states, enabling linear-time sequence modeling and constant-memory autoregressive decoding. Prior work reports a persistent weakness on recall-intensive tasks (Arora et al., 2024): unlike full attention (Vaswani et al., 2017), which retains token-level keys and values for query-dependent retrieval, linear attention must recover relevant information from states that aggregate many historical tokens. Limited recurrent-state capacity is a common explanation for this gap, motivating state expansion (Arora et al., 2024; Zhang et al., 2024; Pan et al., 2026) and efforts to mitigate state collision (Tumma et al., 2026).

A prominent trend is multi-slot routing (Peng et al., 2022; Zhang et al., 2024; Pan et al., 2026; Du et al., 2026): its core idea is to allocate multiple slot-indexed memories and distribute historical updates among them, providing more storage for historical associations than a single shared state. Each query then combines the slot readouts through read routing. This motivates the common interpretation of multi-slot routing as state-capacity expansion. However, this interpretation leaves a critical question unanswered: why do routing designs with equal state capacity—the same number and dimensionality of slots—differ substantially in retrieval performance?

We argue that multi-slot routing **factorizes a query-conditioned selection operator** over historical tokens, rather than merely expanding state capacity: the overlap between historical tokens' write assignments and the current query's read assignments determines how strongly each token is selected. The slot count determines the operator's rank budget, while the routing design determines its usable expressivity. Here, *query-conditioned selectivity* (Fan et al., 2025; Zhang et al., 2026) is the ability to emphasize relevant historical tokens while suppressing distractors; full attention obtains it from a softmax over tokens, and slot routing from competition over slots (Figure 1(a)). The GLA example in Figure 1(b) exposes the selectivity deficit of linear attention: GLA produces a faint peak in the needle region, indicating that it attempts to attend to the needle, but this signal is overwhelmed by weights

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

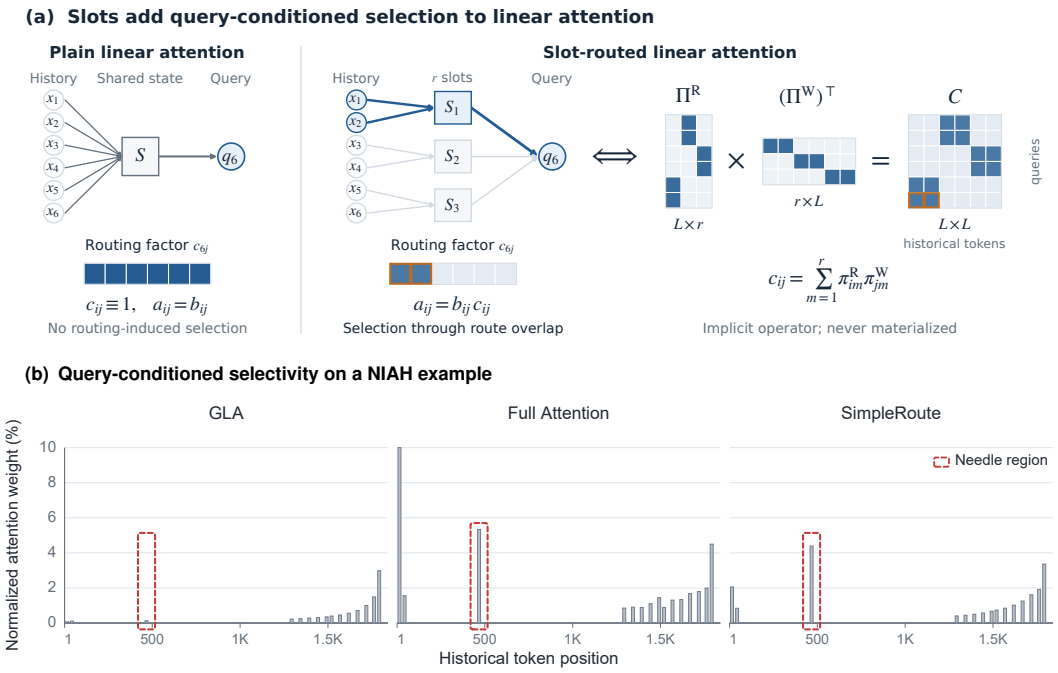

Figure 1: **Query-conditioned selectivity through slot routing.** (a) Plain linear attention has the trivial routing factor $c_{ij} = 1$; slot routing introduces $C = \Pi^{R}(\Pi^{W})^{\top}$, giving $a_{ij} = b_{ij}c_{ij}$. Heatmaps show routing factors before causal masking. (b) Normalized attention weights of the key tokens in a NIAH question over the historical positions, averaged over the key tokens, with the needle region boxed in red: GLA shows only a faint needle peak, whereas SimpleRoute, like full attention (Transformer++), places a prominent peak on it. Averaged over 1,500 S-NIAH cases, SimpleRoute places about $10\times$ more attention on the needle than GLA (Figure 3).

near the end of the history. We formalize this **selectivity view of multi-slot routing** in Section 3.2 and test it in Section 5 by comparing routing designs at fixed or matched state capacity.

This view exposes two structural restrictions in existing designs. In ABC (Peng et al., 2022) and GSA (Zhang et al., 2024), *cross-token selectivity leakage* lets selecting one historical token also amplify other values aggregated into the same slot. In SSE (Pan et al., 2026) and MoM (Du et al., 2026), *tied selectivity* couples write and read routing through a shared routing code (Sections 3.3.1 and 3.3.2; MoM in Section A.2).

Guided by these diagnoses, we propose **SimpleRoute**, a minimal routing design that removes both restrictions (Section 4): it keeps each token's key–value association intact, so selecting one token no longer amplifies other tokens sharing its slot, and it decouples write and read routing, so the slots a token is written to no longer dictate the slots it reads from. As a result, SimpleRoute places a pronounced peak on the needle region in Figure 1(b), approaching the selectivity of full attention (Section 5.3).

Our contributions are as follows:

- A **selectivity view of multi-slot routing** that separates the slot-count rank budget from the routing design's usable expressivity.
- A diagnosis of **cross-token selectivity leakage** in ABC/GSA and **tied selectivity** in SSE/MoM.
- **SimpleRoute**, which removes both restrictions and achieves selectivity approaching that of full attention. At 1B parameters and 50B tokens, it attains the highest commonsense and real-world retrieval averages among the compared recurrent models and improves real-world retrieval by 6.35 points over the strongest same-budget baseline.

2 Related work

Linear attention and recurrent retrieval bottlenecks. Kernelized linear attention enables linear-time training and constant-memory decoding (Katharopoulos et al., 2020; Shen et al., 2021; Choromanski et al., 2021). RetNet and GLA add fixed and data-dependent decay (Sun et al., 2023; Yang et al., 2024a), while DeltaNet, Gated DeltaNet, KDA, and preconditioned variants refine state updates to reduce conflicting associations (Yang et al., 2024b; 2025; Kimi Team, 2025; Tumma et al., 2026). Retrieval nevertheless remains constrained by finite recurrent states (Arora et al., 2024; Qin et al., 2024); SimpleRoute instead studies how these states are selectively written and read.

Multi-slot routing and state capacity. ABC compresses keys and values into memory slots (Peng et al., 2022), GSA adds gated updates (Zhang et al., 2024), SSE uses sparse token-to-state assignments (Pan et al., 2026), MoM conditionally activates multiple memories (Du et al., 2026), and log-linear attention grows the number of states logarithmically with sequence length (Guo et al., 2026). These methods primarily emphasize state capacity or organization; our framework separates token-specific association from routing-induced selectivity, exposing restrictions that state size alone does not explain.

Routing in softmax attention. Routing Transformer and Reformer restrict each query to keys in the same cluster or hash bucket (Roy et al., 2021; Kitaev et al., 2020); in our framework, these are hard routings whose read and write assignments share a codebook or, with Reformer's shared query–key projection, coincide. MoBA and NSA route queries to blocks of past tokens (Lu et al., 2025; Yuan et al., 2025), and MoH routes tokens to heads (Jin et al., 2025). Unlike SimpleRoute, these methods select over retained token-level keys and values rather than fixed-size recurrent states.

Query-conditioned selectivity in linear attention. Mamba makes state updates input-dependent, deciding what to write and forget (Gu & Dao, 2024); we instead study how a query selects among stored tokens when reading. Magnitude-aware linear attention studies how normalization limits its retrieval sharpening (Fan et al., 2025), MHLA addresses context collapse through multi-head partitioning (Zhang et al., 2026), and related analyses connect history compression to reduced expressivity (Han et al., 2024; Mongaras & Larson, 2025). We connect selectivity to multi-slot write/read assignments, whose overlap reweights historical tokens for each query.

3 A Selectivity View of Multi-Slot Routing

3.1 Query-conditioned selectivity and linear attention

Query-conditioned selectivity is the ability to concentrate retrieval on historical tokens relevant to the current query while suppressing distractors (Fan et al., 2025; Zhang et al., 2026). Full attention obtains it from a row-wise softmax over historical tokens, which makes them compete for retrieval mass: the ratio of the attention weights of two tokens j and ℓ is $\exp\big(q_i^\top (k_j - k_\ell)/\sqrt{d}\big)$, so differences in how well they match the query are exponentially amplified.

Linear attention replaces this softmax with a query–key association computed through a feature map ϕ:

$$y_i^{\mathrm{LA}} = \sum_{j \le i} \underbrace{b_{ij}^{\mathrm{LA}}}_{\text{token-specific association}} v_j, \qquad b_{ij}^{\mathrm{LA}} := \phi(q_i)^\top \phi(k_j), \tag{1}$$

and implements it recurrently by accumulating key–value updates into a fixed-size state before the current query arrives. Consequently, b_{ij}^{LA} cannot be normalized competitively over historical tokens: a token-level softmax would require the query before the history is compressed, and the linear normalizer used by some variants divides all associations of a query by the same scalar, leaving their ratios unchanged. The relative weights of historical tokens are therefore determined entirely by the feature-map association, with no competitive stage to amplify them, as reflected in the faint needle peak of GLA in Figure 1(b). Prior work has documented this weakness (Fan et al., 2025; Zhang et al., 2026) but has not connected it to multi-slot routing; we next show that routing supplies precisely the missing competitive mechanism.

3.2 MULTI-SLOT ROUTING AS QUERY-CONDITIONED SELECTION

Multi-slot routing is usually introduced to expand state capacity: it replaces the shared recurrent state with r slot-indexed states of fixed dimensions. The write assignment $\boldsymbol{\pi}_j^{\mathrm{W}} \in \mathbb{R}^r$ distributes historical token j's update across these states, and the read assignment $\boldsymbol{\pi}_i^{\mathrm{R}} \in \mathbb{R}^r$ determines how query i combines their readouts. Let b_{ij}^{MS}, analogous to b_{ij}^{LA}, denote the association between query i and token j within each state. Without prescribing how these quantities are computed, the general output is

$$y_i^{\mathrm{MS}} = \sum_{m=1}^{r} \pi_{im}^{\mathrm{R}} \left(\sum_{j \le i} \pi_{jm}^{\mathrm{W}} b_{ij}^{\mathrm{MS}} v_j \right) = \sum_{j \le i} \underbrace{b_{ij}^{\mathrm{MS}}}_{\text{token-specific association}} \underbrace{\left(\sum_{m=1}^{r} \pi_{im}^{\mathrm{R}} \pi_{jm}^{\mathrm{W}} \right)}_{\text{slot-mediated } \textbf{selection}} v_j. \tag{2}$$

We denote the slot-mediated selection by $c_{ij} := \langle \boldsymbol{\pi}_i^{\mathrm{R}}, \boldsymbol{\pi}_j^{\mathrm{W}} \rangle$ and the resulting effective retrieval weight by $a_{ij}^{\mathrm{MS}} := b_{ij}^{\mathrm{MS}} c_{ij}$. Standard linear attention is recovered with $r = 1$, unit write/read assignments, and $b_{ij}^{\mathrm{MS}} = b_{ij}^{\mathrm{LA}}$, so that $c_{ij} \equiv 1$. Equation (2) thus shows that routing does more than add capacity: it multiplies every token-specific association by a routing factor.

Unlike b_{ij}^{MS}, the routing weights are computed for each token over the r slots, independently of the other tokens. They can therefore be normalized competitively—for example, by a softmax over slots—without breaking the recurrence, reintroducing competition at the level of slots. For two historical tokens j and ℓ, whenever the denominators are nonzero,

$$\frac{a_{ij}^{\mathrm{MS}}}{a_{i\ell}^{\mathrm{MS}}} = \frac{b_{ij}^{\mathrm{MS}}}{b_{i\ell}^{\mathrm{MS}}} \cdot \frac{c_{ij}}{c_{i\ell}}. \tag{3}$$

The first factor is the contrast already available to linear attention; the second is contributed by routing. Because slots compete for each token's write assignment and each query's read assignment, $c_{i\ell}$ can approach zero for tokens written to slots the query does not read, while c_{ij} remains large for tokens in the selected slots. Unlike sharpening the feature map ϕ, which changes associations without making tokens compete, this normalization is zero-sum: shifting routing mass toward the slots a query reads necessarily removes it from the others. Figure 1(a) illustrates the mechanism. On the left, all historical tokens share one state, so $c_{ij} = 1$ leaves the associations unchanged. On the right, q_6 reads the slot that receives x_1 and x_2, giving these tokens larger routing weights than the others. The factorization $C = \Pi^{\mathrm{R}}(\Pi^{\mathrm{W}})^\top$ summarizes this selection without materializing a token-to-token matrix. **An effective multi-slot routing design should therefore make C as expressive as possible within a fixed slot budget.**

The rank budget of slot-mediated selection. One restriction is inherent to any routing relation mediated through r slots. Let $\Pi^{\mathrm{R}}, \Pi^{\mathrm{W}} \in \mathbb{R}^{L \times r}$ collect the read and write assignments, respectively. Then

$$C = \Pi^{\mathrm{R}}(\Pi^{\mathrm{W}})^\top, \qquad \operatorname{rank}(C) \le r. \tag{4}$$

Thus, r sets the rank budget of routing-induced selectivity, while the routing design determines which selection patterns C can express within it (Figure 2); Section 3.3 shows that existing designs impose additional, unnecessary restrictions on C.

3.3 DIAGNOSING EXISTING MULTI-SLOT DESIGNS

3.3.1 ABC/GSA: MISSING TOKEN-SPECIFIC ASSOCIATION AND SELECTIVITY LEAKAGE

ABC (Peng et al., 2022) summarizes keys and values into slots with a shared write assignment $\pi_{jm}^{\mathrm{W,ABC}}$, its memory-control coefficient, and reads the summaries with a softmax over slots. Substituting these assignments into Equation (2) (derivation in Section A.1) gives

$$y_i^{\mathrm{ABC}} = \sum_{j \le i} \underbrace{1}_{b_{ij}^{\mathrm{ABC}}} \underbrace{\sum_{m=1}^{r} \pi_{jm}^{\mathrm{W,ABC}} \left[\operatorname{softmax}(h_i)\right]_m}_{c_{ij}^{\mathrm{ABC}}} v_j, \qquad h_{im} = q_i^\top \widetilde{k}_{i,m} = \sum_{\ell \le i} \pi_{\ell m}^{\mathrm{W,ABC}} q_i^\top k_\ell, \tag{5}$$

where $\widetilde{k}_{i,m}$ is slot m's aggregated key.

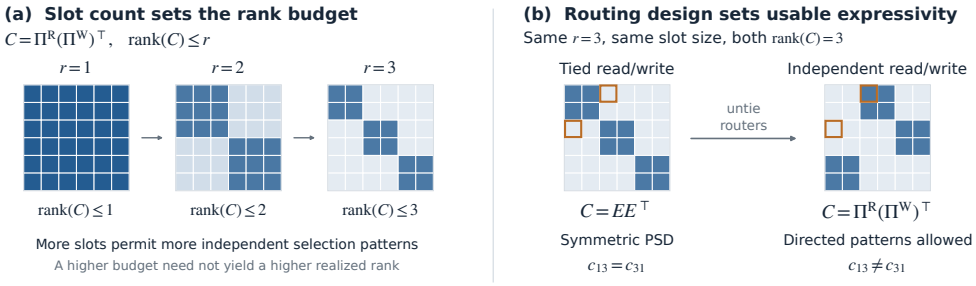

Figure 2: **Slot count sets the rank budget; routing design sets usable expressivity.** (a) More slots raise the upper bound on $\mathrm{rank}(C)$ without guaranteeing a higher realized rank. (b) At the same slot count and realized rank, identical read and write assignments constrain C to a symmetric positive-semidefinite Gram matrix, whereas independent assignments permit directed relations. Heatmaps are illustrative routing factors before causal masking.

Missing token-specific association. Since $b_{ij}^{\mathrm{ABC}} = 1$, ABC retains no token-specific association analogous to b_{ij}^{LA}; tokens are differentiated only through their slot assignments.

Cross-token selectivity leakage. A strong match $q_i^{\top} k_\ell$ raises slot m's read weight through h_{im} and hence the contribution of another token j stored there, even when the query does not match k_j, because the slot weight multiplies all aggregated values without preserving which key supplied the matching evidence. Formally, a design exhibits this *cross-token selectivity leakage* when $\partial c_{ij}/\partial k_\ell \neq 0$ for some $\ell \neq j$. GSA retains this separate key/value organization with recurrent decay, so the same diagnosis applies after absorbing decay into the effective write coefficients (Zhang et al., 2024).

3.3.2 SSE: Preserved token association but tied selectivity

Let $\bar{e}_t \in \mathbb{R}^r$ denote the partition gate of SSE (Pan et al., 2026) after its softmax and top-k mask. SSE preserves the linear-attention association and applies the same gate to the state input and output, so Equation (2) becomes

$$y_i^{\mathrm{SSE}} = \sum_{j \le i} \underbrace{b_{ij}^{\mathrm{LA}}}_{b_{ij}^{\mathrm{SSE}}} \underbrace{\bar{e}_i^{\top} \bar{e}_j}_{c_{ij}^{\mathrm{SSE}}} v_j, \qquad \boldsymbol{\pi}_t^{\mathrm{W,SSE}} = \boldsymbol{\pi}_t^{\mathrm{R,SSE}} = \bar{e}_t. \tag{6}$$

SSE thus retains each token's query–key association and uses token-derived routes rather than pooled slot keys, avoiding cross-token selectivity leakage.

Tied selectivity. Beyond the inherent rank-r budget, SSE forces each token's write and read assignments to be identical, which restricts the routing factor to a Gram matrix:

$$C^{\mathrm{SSE}} = \bar{E}\bar{E}^{\top} = (C^{\mathrm{SSE}})^{\top}, \qquad z^{\top} C^{\mathrm{SSE}} z = \|\bar{E}^{\top} z\|_2^2 \ge 0. \tag{7}$$

This excludes more general rank-r read–write relations in which storing a token and retrieving information for a query require different routing assignments (Figure 2(b)). Because the tie is imposed on the routing assignments themselves, it persists under causal masking, which retains only entries with $j \le i$: each query must read from exactly the slots to which its own token is written, so a query that should retrieve evidence from one slot must also store itself there, and the Gram structure bounds every retained entry by $c_{ij}^2 \le c_{ii} c_{jj}$. MoM (Du et al., 2026) similarly derives write and read routing from a single per-token router output, tying their supports (Section A.2).

Together, these diagnoses motivate a routing design that preserves token-specific association, avoids cross-token selectivity leakage, and separates write and read assignments, making C as expressive as possible within a fixed slot budget.

4 A SIMPLE AND EXPRESSIVE ROUTING DESIGN

4.1 A SIMPLE INSTANTIATION OF MULTI-SLOT ROUTING

Two separately parameterized dot-product softmax routers suffice to remove both restrictions of Section 3.3. Let $q_t, k_t \in \mathbb{R}^d$ and $v_t \in \mathbb{R}^{d_v}$ denote the query, key, and value vectors at position t. We learn r write-slot codes and r read-slot codes, $E^{\mathrm{W}}, E^{\mathrm{R}} \in \mathbb{R}^{r\times d}$, and route

$$\boldsymbol{\pi}_t^{\mathrm{W}} = \mathrm{softmax}\big(E^{\mathrm{W}} k_t\big)\,, \qquad \boldsymbol{\pi}_t^{\mathrm{R}} = \mathrm{softmax}\big(E^{\mathrm{R}} q_t\big)\,, \tag{8}$$

where both softmaxes normalize over the r slots: the key-derived route decides which slots receive the current key–value update, and the query-derived route decides which slots contribute to the current output. Substituting Equation (8) into Equation (2) gives

$$y_i^{\mathrm{SR}} = \sum_{j\le i} \underbrace{\phi(q_i)^\top \phi(k_j)}_{b_{ij}^{\mathrm{SR}}:\ \text{token-specific association}} \underbrace{\big\langle \boldsymbol{\pi}_i^{\mathrm{R}}, \boldsymbol{\pi}_j^{\mathrm{W}} \big\rangle}_{c_{ij}^{\mathrm{SR}}:\ \text{slot-mediated } \textbf{selection}} v_j. \tag{9}$$

This construction **(1)** preserves token-specific association, because scalar write assignments multiply bound outer products $\phi(k_t)v_t^\top$; **(2)** avoids cross-token selectivity leakage, because c_{ij} depends only on the pair (q_i, k_j); and **(3)** separates write and read routing through independent codebooks, allowing directed selection relations.

4.2 LEVERAGING MULTI-HEAD STATES FOR EFFICIENT ROUTING

Multi-head linear attention already maintains H recurrent states, each of which can serve as a routing slot without additional state storage. More generally, we assign m slots to each head, each with a recurrent state of the original per-head shape, giving $r = mH$ slots; $\mathcal{R}_h$ denotes those of head h. Routing is computed within each head and normalized globally: for $\rho \in \mathcal{R}_h$, $\pi_{t\rho}^{\mathrm{W}} \propto \exp\big((e_\rho^{\mathrm{W}})^\top k_{h,t} + \beta_\rho^{\mathrm{W}}\big)$ and $\pi_{t\rho}^{\mathrm{R}} \propto \exp\big((e_\rho^{\mathrm{R}})^\top q_{h,t} + \beta_\rho^{\mathrm{R}}\big)$, with slot codes $e_\rho^{\mathrm{W}}, e_\rho^{\mathrm{R}} \in \mathbb{R}^{d_k}$, scalar biases $\beta_\rho^{\mathrm{W}}, \beta_\rho^{\mathrm{R}}$, per-head query/key dimension d_k, and softmax over all r slots. On a Gated Linear Attention (GLA) backbone (Yang et al., 2024a), each slot retains head h's forget gate $\alpha_{h,t}$, while the routes modulate only writes and reads:

$$S_{\rho,t} = \alpha_{h,t} \odot S_{\rho,t-1} + \pi_{t\rho}^{\mathrm{W}}\, \phi(k_{h,t}) v_{h,t}^\top, \qquad y_{h,t} = \sum_{\rho\in\mathcal{R}_h} \pi_{t\rho}^{\mathrm{R}}\, \phi(q_{h,t})^\top S_{\rho,t}. \tag{10}$$

Because all slots in $\mathcal{R}_h$ share head h's gate, the cumulative decay is absorbed into $b_{ij} = \phi(q_{h,i})^\top\big(\gamma_{j\to i}^h \odot \phi(k_{h,j})\big)$, with $\gamma_{j\to i}^h = \prod_{t=j+1}^i \alpha_{h,t}$, leaving c_{ij} unchanged.

We use two regimes. **State-neutral routing ($r = H$)** uses one slot per head, so the recurrent states match standard multi-head GLA in number and shape, and routing strengthens selectivity without expanding state capacity. Each head's routing factor then reduces to $c_{ij} = \pi_{ih}^{\mathrm{R}} \pi_{jh}^{\mathrm{W}}$, akin to head-level routing (Jin et al., 2025) but applied to both writes and reads; because both weights are softmax-normalized across heads, heads compete for each token's update and each query's readout rather than being gated independently. **Expanded routing ($r > H$)** increases both state capacity and the rank budget of C; we use it to evaluate the method's full performance, without attributing its gains exclusively to selectivity.

Routing adds little overhead to the underlying linear attention: $2r(d_k + 1)$ parameters per layer (99,072 in total, or 0.03% of the model, at $r = 16$) and $O(Lrd_k)$ token-local computation. Beyond the intended m-fold state in the expanded regime, decoding cost remains independent of context length (Section C).

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

We instantiate SimpleRoute on GLA (Yang et al., 2024a) and train all models from scratch on SlimPajama-627B (Soboleva et al., 2023) at 2K context, following prior protocols (Yang et al., 2024a; Du et al., 2026; Tumma et al., 2026). Our main setting trains 1B-parameter models on 50B tokens with

Table 1: Main results, grouped by model scale and training budget. Real-world retrieval, the key weakness of linear attention, is listed first; RW Avg. averages the four retrieval tasks and Common Avg. averages six commonsense reasoning tasks. Best and second-best scores are bold and underlined within each block, among recurrent models only (Transformer++ is excluded). †External results with 1.3B parameters and 100B training tokens (2× our budget), included as references rather than controlled comparisons. ‡MoM is substantially larger than the other 340M-scale models (512M total, 411M activated parameters). Parameter counts marked ≈ are nominal sizes reported by the source papers. Sources are in Section B.2.

Model	Params (M)	Real-world retrieval ↑					Language modeling ↓		Commonsense ↑
		FDA	SWDE	SQuAD	DROP	**RW Avg.**	Wiki ppl	LMB. ppl	Common Avg.
1B-scale models / 50B training tokens / 0.5M batch tokens (†1.3B models / 100B tokens)									
GSA†	≈1,300	23.60	29.80	36.00	20.90	27.58	16.70	12.60	<u>51.68</u>
MoM†	≈1,300	**41.14**	34.30	37.08	<u>21.03</u>	<u>33.39</u>	<u>16.64</u>	14.83	50.97
GDN	≈1,000	19.69	49.86	37.70	20.50	31.94	18.05	13.46	50.50
PGDN	≈1,000	18.33	<u>50.32</u>	<u>41.19</u>	20.80	32.66	18.01	13.09	50.84
KDA	≈1,000	23.23	50.14	37.00	19.70	32.52	17.95	14.52	50.01
PKDA	≈1,000	21.69	48.79	38.61	20.70	32.45	17.82	**11.86**	51.31
SimpleRoute	995.042	<u>29.04</u>	**62.83**	**42.23**	**21.95**	**39.01**	**16.29**	<u>12.54</u>	**52.61**
340M-scale models / 15B training tokens									
Transformer++	341.100	25.68	53.65	36.56	20.51	34.10	23.97	26.42	43.93
RetNet	341.106	5.90	9.28	22.41	18.59	14.05	31.07	87.11	40.19
HGRN2	341.149	11.53	17.34	24.08	17.35	17.58	27.90	77.40	41.46
GLA	341.708	8.53	31.68	<u>32.10</u>	19.89	23.05	26.32	26.18	<u>43.69</u>
GSA	322.250	6.36	16.87	21.90	16.72	15.46	28.17	82.50	40.67
GSA-s256	347.416	7.35	<u>35.01</u>	30.53	16.48	22.34	<u>25.27</u>	34.17	43.29
MoM‡	512/411	**22.98**	29.90	29.69	**20.99**	<u>25.89</u>	25.86	55.41	41.97
DN	341.594	8.53	27.09	28.32	17.80	20.44	29.04	45.76	41.20
PDN	341.988	14.16	29.17	28.89	16.40	22.16	29.05	43.05	41.68
GDN	341.791	13.61	29.43	29.83	17.20	22.52	27.08	31.39	42.73
PGDN	342.185	<u>14.25</u>	32.24	31.76	17.40	23.91	26.86	28.43	43.59
KDA	354.226	13.88	34.11	29.69	16.90	23.65	26.18	31.37	42.75
PKDA	354.620	<u>14.25</u>	34.65	31.39	18.00	24.57	25.98	<u>25.33</u>	43.55
SimpleRoute	341.807	12.43	**37.89**	**34.42**	<u>19.93</u>	**26.17**	**24.27**	**24.64**	**44.33**

a batch size of 0.5M tokens, matching the data, context length, model scale, and token/batch budgets of the recent recurrent baselines reported by Tumma et al. (2026). The 340M-parameter setting trains on 15B tokens; all in-house models in this setting share the same data, tokenizer, token budget, batch size, and optimization recipe. By default, SimpleRoute retains GLA's key/value dimensions and uses $r = 64$ routing slots at 1B and $r = 16$ at 340M. Unless otherwise stated, our analyses and ablations use the 340M setting. Model, training, and evaluation details are in Section B.1.

5.2 Main results

Table 1 compares SimpleRoute with full-attention and recent recurrent baselines at both scales.

State-of-the-art recurrent performance. In the main 1B setting, SimpleRoute attains the highest real-world retrieval and commonsense averages and the lowest WikiText perplexity among all compared recurrent models, leading the same-budget baselines on 7 of 8 metrics and trailing only PKDA on LAMBADA perplexity. It also surpasses the 1.3B GSA and MoM on both averages despite their twice-larger training budgets. At the smaller scale, it likewise ranks first among recurrent models on both perplexities and on the commonsense and retrieval averages; Transformer++ leads retrieval and WikiText perplexity at this scale but trails SimpleRoute on LAMBADA perplexity and commonsense reasoning.

Table 2: Effect of routing. $r = mH$ is the total number of routing slots across $H = 4$ heads; state capacity is normalized to GLA. Models are trained at 2K, with 4K testing length extrapolation.

Model	Params	State	Wiki	LMB.	Common	RW	S-NIAH Avg. ↑			
	(M)	capacity	ppl ↓	ppl ↓	Avg.-6 ↑	Avg.-4 ↑	@1K	@2K	@4K	All
GLA	341.708	1×	26.32	26.18	43.69	23.05	87.20	49.20	15.53	50.64
SimpleRoute ($r = 4$)	341.733	1×	24.81	28.30	44.35	21.05	80.33	58.87	22.60	53.93
SimpleRoute ($r = 8$)	341.757	2×	24.39	27.13	44.20	27.96	86.27	69.40	36.00	63.89
SimpleRoute ($r = 16$)	341.807	4×	24.27	**24.64**	44.33	26.17	**97.67**	80.13	28.00	68.60
SimpleRoute ($r = 64$)	342.104	16×	**23.78**	25.52	**44.65**	**29.68**	89.20	**85.67**	**59.60**	**78.16**

Largest gains on retrieval. Retrieval, widely regarded as the key weakness of linear attention (Arora et al., 2024), is where SimpleRoute gains most. At 1B, it improves the retrieval average by 6.35 points over the strongest same-budget baseline (12.51 points on SWDE), versus 1.30 points on commonsense reasoning, and exceeds the 1.3B GSA and MoM by 11.43 and 5.62 points. At the smaller scale, it exceeds GSA by 10.71 points, MoM by 0.28 points despite MoM's 50% more total parameters, and the capacity-matched GSA-s256 by 3.83 points, suggesting that the gains come from the routing design rather than from state capacity alone.

Efficiency. Routing preserves the length scaling of linear attention (Section C). From 2K to 16K context, GLA and SimpleRoute retain 98.4–99.3% of their training throughput, whereas Transformer++ retains 46.8%; at 16K, SimpleRoute ($r = 16$) trains 1.83× faster than Transformer++ and reduces decoding FLOPs per token and peak decoding memory by 67.6% and 68.6%, with both costs nearly constant in context length.

5.3 Measuring query-conditioned selectivity

To examine whether these retrieval gains reflect stronger query-conditioned selectivity, we measure where each model attends when the question refers to the needle. Each S-NIAH case consists of a context followed by a question of the form "What is the value of [key]?"; for each token of [key] in the question, we compute the fraction of its attention over the preceding tokens that falls on the needle sentence. For the gated linear models, we take the absolute read coefficients (Ali et al., 2025; Zimerman et al., 2025) and normalize them to sum to one over the preceding tokens, making them comparable to softmax attention; we then average over layers and key tokens (Section B.4). Averaged over 1,500 cases from the three S-NIAH tasks (Figure 3), GLA places only 0.44% of its attention on the needle sentence, whereas SimpleRoute places 4.56%, a roughly tenfold increase, and Transformer++ places 8.07%. Routing thus recovers more than half of full attention's needle-focused weighting. The capacity-matched GSA-s256 also raises needle attention (3.31%) but remains 27% below SimpleRoute, consistent with the cross-token selectivity leakage diagnosed in Section 3.3.1. Figure 1(b) illustrates a single case: GLA's faint needle peak is overwhelmed by recent tokens, whereas SimpleRoute, like Transformer++, places a distinct peak on the needle. These measurements do not by themselves isolate routing from state capacity; Sections 5.4 and 5.5 do.

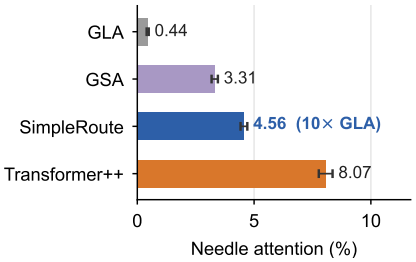

Figure 3: Attention on the needle sentence in S-NIAH (mean over 1,500 cases from the three tasks; error bars: 95% CI).

5.4 Effect of routing

Effect of selectivity alone. With $r = H = 4$ ($m = 1$), SimpleRoute operates in the state-neutral regime of Section 4.2: every head state keeps its original shape, so routing strengthens selectivity without expanding state capacity. In this regime, SimpleRoute improves WikiText perplexity from 26.32 to 24.81, average commonsense accuracy from 43.69 to 44.35, and overall S-NIAH from 50.64 to 53.93, although LAMBADA perplexity and average real-world retrieval do not improve.

Strengthening selectivity alone thus already improves WikiText perplexity, average commonsense accuracy, and overall S-NIAH, consistent with the selectivity view.

Effect of routing scale. Increasing r improves WikiText perplexity and overall S-NIAH across the tested configurations. At $r = 64$, SimpleRoute also achieves the highest commonsense and real-world retrieval averages and the strongest 2K/4K S-NIAH results, while LAMBADA perplexity and 1K S-NIAH remain best at $r = 16$. The gains are largest under length extrapolation: at 4K, $r = 64$ raises S-NIAH-2 from 28.4 to 98.0 and S-NIAH-3 from 7.2 to 55.4 over GLA (Table 9). Because r increases both recurrent-state capacity and the rank budget of C, this sweep evaluates their joint scaling. At a total state size similar to $r = 16$, enlarging the per-slot states (with $r = 8$) further improves the retrieval average from 26.17 to 27.55, suggesting that routing is complementary to wider states (Section B.3), albeit with 11.3% more parameters.

5.5 Validating routing expressivity restrictions

We test the two routing expressivity restrictions diagnosed in Section 3.3 with controlled variants of SimpleRoute. *Unbound* separately aggregates slot-level keys and values, reproducing the loss of token-level binding and cross-token selectivity leakage of ABC/GSA. *Exact-Tied* instead preserves bound updates but uses the same assignment for writing and reading, imposing the tied-selectivity restriction of SSE. *Full SimpleRoute* removes both restrictions by retaining bound key–value updates and separate write/read assignments. The three variants are trained at the 110M scale with the same recipe and are nearly matched in parameters and recurrent-state size.

Table 3: Controlled tests of the routing expressivity restrictions diagnosed in Section 3.3. All models have approximately 110M parameters and use $r = 16$ routing slots.

Variant	Params (M)	State per layer	Wiki ppl ↓	LMB. ppl ↓	Common. Avg.-6 ↑	FDA ↑	SWDE ↑	SQuAD ↑	DROP ↑	RW Avg.-4 ↑
Full SimpleRoute	112.258	204,800	**32.49**	**52.19**	40.39	**13.34**	**23.85**	28.92	**16.82**	**20.73**
Unbound	112.277	204,480	36.93	101.94	38.25	1.36	10.71	20.84	14.28	11.80
Exact-Tied	112.235	204,800	33.28	54.42	**40.51**	4.08	23.76	**29.73**	16.72	18.57

Both restrictions hurt retrieval and language modeling. Unbound causes a large degradation, supporting the importance of preserving key–value binding to avoid cross-token selectivity leakage. Exact-Tied produces smaller declines in both perplexities and the retrieval average; the retrieval decline comes mainly from FDA, while the other retrieval tasks and commonsense accuracy change little. This supports an additional benefit from independent read/write routing, although a smaller one than that of preserving key–value binding.

6 Conclusion

We presented a selectivity view of multi-slot linear attention: routing factorizes a query-conditioned selection operator whose rank budget is set by slot count and whose usable expressivity depends on routing design. This view reveals cross-token selectivity leakage and tied selectivity, motivating SimpleRoute's bound key–value updates and independent write/read routers. At 1B parameters, SimpleRoute achieves the strongest retrieval among the compared recurrent models while retaining linear-time training and constant-memory decoding. Capacity-matched comparisons, routing ablations, and needle-region attention mass indicate that routing design, rather than state size alone, contributes to these gains.

AI Use Statement

We used large language models as assistive tools for manuscript drafting and revision; code development and debugging; executing and managing experiments; preparing scientific figures and conceptual illustrations; developing and checking mathematical derivations; and searching, summarizing, and organizing related literature. These tools supported the research workflow under author

supervision. The authors checked the mathematical arguments, tested the implementation, verified reported results against experimental outputs, and checked citations against the original sources. The authors reviewed the AI-assisted text and figures and retain responsibility for the final manuscript, theoretical claims, implementation, experimental results, figures, and references.

REFERENCES

Ameen Ali, Itamar Zimerman, and Lior Wolf. The hidden attention of Mamba models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1516–1534, 2025.

Simran Arora, Brandon Yang, Sabri Eyuboglu, Avanika Narayan, Andrew Hojel, Immanuel Trummer, and Christopher Ré. Language models enable simple systems for generating structured views of heterogeneous data lakes. *Proceedings of the VLDB Endowment*, 17(2):92–105, 2023. doi: 10.14778/3626292.3626294. URL https://www.vldb.org/pvldb/vol17/p92-arora.pdf.

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, James Zou, Atri Rudra, and Christopher Ré. Simple linear attention language models balance the recall-throughput tradeoff. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 1763–1840. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/arora24a.html.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457, 2018.

Jusen Du, Weigao Sun, Disen Lan, Jiaxi Hu, Tao Zhang, and Yu Cheng. MoM: Linear sequence modeling with mixture-of-memories. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/ae2dd45fb3d7ce1dce45e5de4bdd9ba0-Paper-Conference.pdf.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2368–2378, 2019. doi: 10.18653/v1/N19-1246. URL https://aclanthology.org/N19-1246/.

Qihang Fan, Huaibo Huang, Yuang Ai, and Ran He. Rectifying magnitude neglect in linear attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 21505–21514, 2025.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 12 2023. URL https://zenodo.org/records/10256836.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First conference on language modeling*, 2024.

Han Guo, Songlin Yang, Tarushii Goel, Eric P. Xing, Tri Dao, and Yoon Kim. Log-linear attention. In *The Fourteenth International Conference on Learning Representations*, 2026.

Dongchen Han, Yifan Pu, Zhuofan Xia, Yizeng Han, Xuran Pan, Xiu Li, Jiwen Lu, Shiji Song, and Gao Huang. Bridging the divide: Reconsidering softmax and linear attention. In *NeurIPS*, 2024.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=kIoBbc76Sy.

Peng Jin, Bo Zhu, Li Yuan, and Shuicheng Yan. MoH: Multi-head attention as mixture-of-head attention. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 28233–28255, 2025.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pp. 5156–5165. PMLR, 2020.

Kimi Team. Kimi linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025. URL https://arxiv.org/abs/2510.26692.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020.

Colin Lockard, Prashant Shiralkar, and Xin Luna Dong. OpenCeres: When open information extraction meets the semi-structured web. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 3047–3056, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1309. URL https://aclanthology.org/N19-1309/.

Enzhe Lu, Zhejun Jiang, et al. MoBA: Mixture of block attention for long-context LLMs. In *Advances in Neural Information Processing Systems*, 2025.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

Gabriel Mongaras and Eric C. Larson. On the expressiveness of softmax attention: A recurrent neural network perspective. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=PHcITOi3vV.

Yuqi Pan, Yongqi An, Zheng Li, Yuhong Chou, Rui-Jie Zhu, Xiaohui Wang, Mingxuan Wang, Jinqiao Wang, and Guoqi Li. Scaling linear attention capacity with sparse state expansion. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=R6DrJ4tnGV.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc-Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 1525–1534, 2016.

Hao Peng, Jungo Kasai, Nikolaos Pappas, Dani Yogatama, Zhaofeng Wu, Lingpeng Kong, Roy Schwartz, and Noah A. Smith. ABC: Attention with bounded-memory control. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7469–7483, 2022. doi: 10.18653/v1/2022.acl-long.515. URL https://aclanthology.org/2022.acl-long.515/.

Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. HGRN2: Gated linear RNNs with state expansion. In *Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=y6SqbJfCSk.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 784–789, 2018.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 3531–3539, 2021.

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, June 2023. URL https://huggingface.co/datasets/cerebras/SlimPajama-627B.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Neehal Tumma, Noel Loo, and Daniela Rus. Preconditioned DeltaNet: Curvature-aware sequence modeling for linear recurrences. *arXiv preprint arXiv:2604.21100*, 2026. URL https://arxiv.org/abs/2604.21100.

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.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In *International Conference on Machine Learning*, 2024a.

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

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=r8H7xhYPwz.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23078–23097, 2025.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4791–4800, 2019.

Kewei Zhang, Ye Huang, Yufan Deng, Jincheng YU, Junsong Chen, Huan Ling, Enze Xie, and Daquan Zhou. MHLA: Restoring expressivity of linear attention via token-level multi-head. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=340QjF3jJP.

Yu Zhang, Songlin Yang, Rui-Jie 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 *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=jY4PhQibmg.

Itamar Zimerman, Ameen Ali, and Lior Wolf. Explaining modern gated-linear RNNs via a unified implicit attention formulation. In *The Thirteenth International Conference on Learning Representations*, 2025.

A Additional details for the selectivity analysis

A.1 ABC controller and framework derivation

ABC-MLP (Peng et al., 2022) implements its write control with a learned projection $W_\varphi \in \mathbb{R}^{r\times d}$. For token j and slot m, it first computes a positive score and then normalizes it over the causal prefix for that slot:

$$\alpha_{jm} = \exp((W_\varphi x_j)_m), \qquad \pi^{\mathrm{W,ABC}}_{jm} = \frac{\alpha_{jm}}{\sum_{\ell\le j}\alpha_{\ell m}}. \tag{11}$$

Thus, each row of W_φ acts as a content-based pseudo-query for one slot; the normalization is performed over historical tokens separately for each slot, not across slots for a single token. ABC applies the resulting control to both members of each key–value pair, producing the slot summaries $\widetilde{k}_{i,m} = \sum_{j\le i}\pi^{\mathrm{W,ABC}}_{jm}k_j$ and $\widetilde{v}_{i,m} = \sum_{j\le i}\pi^{\mathrm{W,ABC}}_{jm}v_j$ used in Equation (5), and reads them with $\pi^{\mathrm{R,ABC}}_{im} = \mathrm{softmax}_m(q_i^\top \widetilde{k}_{i,m})$. This gives

$$\begin{aligned} y^{\mathrm{ABC}}_i &= \sum_{m=1}^{r}\pi^{\mathrm{R,ABC}}_{im}\sum_{j\le i}\pi^{\mathrm{W,ABC}}_{jm}v_j \\ &= \sum_{j\le i}\left(\sum_{m=1}^{r}\pi^{\mathrm{R,ABC}}_{im}\pi^{\mathrm{W,ABC}}_{jm}\right)v_j, \end{aligned} \tag{12}$$

which is Equation (2) with $b^{\mathrm{ABC}}_{ij} = 1$ and $c^{\mathrm{ABC}}_{ij} = \sum_m \pi^{\mathrm{R,ABC}}_{im}\pi^{\mathrm{W,ABC}}_{jm}$, as stated in Equation (5).

A.2 MoM as support-tied routing

MoM (Du et al., 2026) computes a single sparse router output g_t for each token. Its top-k support determines which memories receive the token's update, whereas the values of the same g_t determine how the current query combines the memory outputs. Let $z_t = \mathrm{supp}(g_t)$ denote the corresponding binary write activation. At the routing level, MoM therefore instantiates

$$\boldsymbol{\pi}^{\mathrm{R}}_t = g_t, \qquad \boldsymbol{\pi}^{\mathrm{W}}_t = z_t, \qquad c^{\mathrm{MOM}}_{ij} = g_i^\top z_j, \qquad C^{\mathrm{MOM}} = GZ^\top. \tag{13}$$

Consequently, $\mathrm{supp}(\boldsymbol{\pi}^{\mathrm{R}}_t) = \mathrm{supp}(\boldsymbol{\pi}^{\mathrm{W}}_t)$: one routing code must decide both where a token is stored and where it retrieves from. MoM thus has a *support-tied selectivity* restriction analogous to SSE's role coupling, although it is weaker than SSE's weight tying. In particular, because $z_t \neq g_t$ in general, $C^{\mathrm{MOM}} = GZ^\top$ need not be symmetric or positive semidefinite. MoM additionally uses memory-specific key/value projections and an always-active shared memory; these enrich its token-level operator but do not remove the coupling between read and write supports induced by the shared router.

B Additional experimental details

B.1 Main 2K experiments: models, training, and evaluation

The following configurations apply to the 2K experiments in Section 5; the complete training configuration is summarized in Tables 6 and 7.

Models and notation. We instantiate SimpleRoute on Gated Linear Attention (GLA) (Yang et al., 2024a). The 340M models have 24 layers, hidden size 1024, and $H = 4$ heads; the 1B model has 24 layers, hidden size 1792, and the same number of heads. Following Section 4.2, m denotes the number of routed states per head and $r = mH$ their total number. **SimpleRoute** keeps GLA's original key and value dimensions and only increases the number of routed states, providing a clean evaluation of the proposed routing mechanism with negligible parameter overhead. **SimpleRoute-Wide** additionally enlarges the key and value dimensions to test whether routing and increased per-state capacity are complementary. It uses fewer routed states than SimpleRoute ($m = 2$ versus $m = 4$), leaving the two configurations with approximately the same recurrent-state size. Its results are reported in Section B.3.

Training. We follow the common training setup of prior work (Yang et al., 2024a; Du et al., 2026; Tumma et al., 2026). The 340M and 1B models are trained from scratch on 15B- and 50B-token SlimPajama subsets (Soboleva et al., 2023), respectively. Both use the `mistralai/Mistral-7B-v0.1` tokenizer, a 32K vocabulary, 2K context, bf16 precision, and one node with 8 NVIDIA H20 GPUs. Each optimizer update contains 256 sequences (524,288 tokens). The 340M and 1B runs use 28,612 and 95,368 updates, respectively, with random seed 42. We use fused AdamW with peak learning rate 10^{-3}, $(\beta_1, \beta_2) = (0.9, 0.95)$, weight decay 0.1, gradient clipping at 1.0, cosine decay, and 954 warmup steps. All in-house comparisons within the same scale share the data, tokenizer, context length, token budget, batch size, and optimization recipe.

Evaluation. To evaluate language modeling, we report perplexity on WikiText (Merity et al., 2017) and LAMBADA (Paperno et al., 2016). To measure general zero-shot capability, we report accuracy on LAMBADA, PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC-Easy, and ARC-Challenge (Clark et al., 2018) using lm-evaluation-harness (Gao et al., 2023). To assess retrieval, we use the real-world FDA information extraction (Arora et al., 2023; 2024), SWDE (Lockard et al., 2019), SQuAD (Rajpurkar et al., 2018), and DROP (Dua et al., 2019) tasks with 2K inputs, together with the three RULER S-NIAH tasks (Hsieh et al., 2024) at 1K, 2K, and 4K. The 4K setting evaluates length extrapolation beyond the 2K training context.

B.2 Baseline sources for the main comparison

Main results at 2K context. Baselines are Transformer++ (Touvron et al., 2023), RetNet (Sun et al., 2023), HGRN2 (Qin et al., 2024), GLA (Yang et al., 2024a), GSA (Zhang et al., 2024), MoM (Du et al., 2026), DN (Yang et al., 2024b), PDN (Tumma et al., 2026), GDN (Yang et al., 2025), PGDN (Tumma et al., 2026), KDA (Kimi Team, 2025), and PKDA (Tumma et al., 2026). Results for DN, PDN, GDN, PGDN, KDA, and PKDA are taken from Tumma et al. (2026). Results for RetNet, HGRN2, GSA, and MoM are taken from the setting labeled *380M* in Du et al. (2026). Transformer++, GLA, and GSA-s256 are our implementations. The original GSA uses a narrower state dimension, so we implement GSA-s256 with the same state dimension and recurrent-state capacity as SimpleRoute. MoM parameters denote total/activated parameters in our table.

For the 1B-scale block, GDN, PGDN, KDA, and PKDA results are from Tumma et al. (2026), whose configuration uses 50B training tokens and 0.5M batch tokens. Our SimpleRoute checkpoint uses the same nominal model scale and token/batch budgets, with $r = 64$ routing slots and 995.042M parameters. The 1.3B/100B GSA and MoM results are from Zhang et al. (2024) and Du et al. (2026), respectively. These nominal scale labels follow the source papers and do not denote exact parameter counts, and the 1.3B models are external references rather than matched-budget comparisons. For all rows, Common Avg. is computed over six commonsense tasks (ARC-Easy, ARC-Challenge, HellaSwag, LAMBADA accuracy, PIQA, and WinoGrande) and RW Avg. over the four retrieval tasks shown in the table, rather than the larger task suites in some source papers.

B.3 Wider per-slot states

The default SimpleRoute keeps GLA's key and value dimensions and increases only the number of routing slots. To test whether routing is complementary to larger per-slot states, SimpleRoute-Wide enlarges the key and value dimensions and uses $r = 8$ ($m = 2$) routing slots, so that its total recurrent-state size approximately matches that of the default $r = 16$ configuration. Table 4 compares it with GLA and the default configuration under the same 15B-token recipe. SimpleRoute-Wide further

improves both perplexities and the commonsense and real-world retrieval averages, suggesting that routing and larger per-slot states are complementary, although it also uses 11.3% more parameters than GLA.

Table 4: SimpleRoute with wider per-slot states at the 340M scale.

Model	Params (M)	Language modeling		Commonsense reasoning							Real-world retrieval				
		Wiki ppl ↓	LMB. ppl ↓	ARC-E	ARC-Cn	Hella-n	LMB. acc	PIQA	Wino.	Common Avg. ↑	FDA	SWDE	SQuAD	DROP	RW Avg. ↑
GLA	341.708	26.32	26.18	46.80	24.74	36.57	36.87	65.13	52.01	43.69	8.53	31.68	32.10	19.89	23.05
SimpleRoute ($r = 16$)	341.807	24.27	24.64	47.39	23.63	38.35	37.32	66.59	52.72	44.33	12.43	37.89	34.42	19.93	26.17
SimpleRoute-Wide ($r = 8$)	380.424	23.67	24.34	48.99	25.17	38.95	37.03	66.59	51.78	44.75	14.16	40.50	35.46	20.08	27.55

B.4 Needle-region attention mass

Section 5.3 and Figure 1(b) use the GLA, GSA-s256, SimpleRoute ($r = 16$), and Transformer++ models of the 340M setting; GSA in Figure 3 denotes the capacity-matched GSA-s256. Each S-NIAH case c consists of a context containing a needle sentence, followed by a question of the form "What is the value of [key]?". Let $\mathcal{Q}_c$ be the positions of the [key] tokens in the question and $\mathcal{G}_c$ the positions of the needle sentence in the context; the key tokens carry the retrieval cue, so their attention indicates where the model looks for the answer. Let $w_{ij}^{(m,c,\ell)}$ be the coefficient with which layer ℓ of model m reads token j at query position i. For the gated linear models, whose read coefficients are signed, we first take element-wise absolute values, following the implicit-attention analyses of gated linear models (Ali et al., 2025; Zimerman et al., 2025), so that negative coefficients also count as attention while zero coefficients remain zero. We then normalize the absolute coefficients to sum to one over the preceding tokens within each layer, which preserves the ratios between tokens and yields a distribution directly comparable to softmax attention. Finally, we average the normalized weights over the N_L layers and the needle-sentence mass over the key tokens:

$$p_{ij}^{(m,c)} = \frac{1}{N_L}\sum_{\ell=1}^{N_L} \frac{\left|w_{ij}^{(m,c,\ell)}\right|}{\sum_{k<i}\left|w_{ik}^{(m,c,\ell)}\right|}, \qquad M_c^{(m)} = \frac{100}{|\mathcal{Q}_c|}\sum_{i\in\mathcal{Q}_c}\sum_{j\in\mathcal{G}_c} p_{ij}^{(m,c)}, \tag{14}$$

where the denominator covers all tokens before the query position and excludes the query itself. We report the mean of $M_c^{(m)}$ over the $C = 1{,}500$ cases from the three S-NIAH tasks (500 per task), that is, the average of per-case ratios rather than a pooled ratio. Within each layer, the coefficients are

$$w_{ij}^{\mathrm{GLA}} = \sum_h b_{ij}^h, \qquad w_{ij}^{\mathrm{SR}} = \sum_h b_{ij}^h \sum_{\rho\in\mathcal{R}_h} \pi_{i\rho}^{\mathrm{R}}\pi_{j\rho}^{\mathrm{W}}, \qquad w_{ij}^{\mathrm{T{+}{+}}} = \frac{1}{H}\sum_h \mathrm{softmax}_{k\le i}\left(\frac{(q_i^h)^\top k_k^h}{\sqrt{d_k}}\right)_j, \tag{15}$$

where b_{ij}^h is head h's query–key association including GLA's cumulative forget-gate decay, $b_{ij}^h = \phi(q_{h,i})^\top\big(\gamma_{j\to i}^h \odot \phi(k_{h,j})\big)$ with $\gamma_{j\to i}^h = \prod_{t=j+1}^i \alpha_{h,t}$, so that w_{ij} is the coefficient with which the query actually reads token j's value. For GSA-s256, w_{ij} is its implicit read coefficient c_{ij}^{GSA} from Equation (5), with the recurrent decay absorbed into the write assignments. For Transformer++, the softmax is computed within each head before averaging over heads; the query's own weight is then excluded and the remaining weights are renormalized as in $p_{ij}^{(m,c)}$. Table 5 reports the mean, standard deviation, and bootstrap 95% confidence interval of $M_c^{(m)}$, and Figure 1(b) plots the layer-averaged normalized weights $p_{ij}^{(m,c)}$, averaged over the key tokens $i \in \mathcal{Q}_c$, for a single randomly selected case.

Table 5: Needle-sentence attention mass on S-NIAH ($C = 1{,}500$ cases from the three tasks).

Model	Mean (%)	Std.	95% CI (bootstrap)
GLA	0.44	0.87	[0.40, 0.49]
GSA-s256	3.31	2.76	[3.18, 3.45]
SimpleRoute	4.56	2.84	[4.42, 4.71]
Transformer++	8.07	5.99	[7.77, 8.36]

B.5 Training and optimization details

Tables 6 and 7 report the 2K training configurations and optimizer hyperparameters.

Table 6: Training configurations at 2K context.

Item	340M	1B
Dataset	`SlimPajama-15B`	`SlimPajama-50B`
Tokenizer	`mistralai/Mistral-7B-v0.1`	`mistralai/Mistral-7B-v0.1`
Vocabulary size	32,000	32,000
Random seed	42	42
Precision	bf16	bf16
Hardware	1 node, 8 H20 GPUs	1 node, 8 H20 GPUs
Context length	2048	2048
Per-GPU micro-batch size	16	8
Gradient accumulation steps	2	4
Global batch size	256 sequences/update	256 sequences/update
Tokens per optimizer update	524,288	524,288
Training steps	28,612	95,368
Total training tokens	15B	50B
Warmup steps	954	954
Warmup tokens	0.5B	0.5B

Table 7: Optimization hyperparameters.

Hyperparameter	340M	1B
Optimizer	fused AdamW (PyTorch)	fused AdamW (PyTorch)
Learning rate	1×10^{-3}	1×10^{-3}
Learning rate scheduler	cosine	cosine
Weight decay	0.1	0.1
Gradient clipping	1.0	1.0
Adam β_1/β_2	0.9 / 0.95	0.9 / 0.95
Checkpoint interval	4,000 steps	4,000 steps
Logging interval	20 steps	20 steps

B.6 Model scale configurations

Table 8 lists the model configurations used in our experiments: 110M for the controlled tests in Section 5.5, and 340M and 1B for the remaining experiments.

Table 8: Model configurations. All models use 4 attention heads and 2K context.

Model scale	Heads	Layers	Hidden size	Tokens
110M	4	18	640	15B
340M	4	24	1024	15B
1B	4	24	1792	50B

B.7 Full S-NIAH results

Table 9 reports the complete results for all three RULER S-NIAH variants at every evaluated context length.

Table 9: Full zero-shot performance (%) on RULER S-NIAH (Hsieh et al., 2024). All models are trained at 2K; the 4K columns evaluate length extrapolation.

	S-NIAH-1			S-NIAH-2			S-NIAH-3			Average			
Model	1K	2K	4K	1K	2K	4K	1K	2K	4K	1K	2K	4K	All
GLA	92.4	30.0	11.0	99.4	83.4	28.4	69.8	34.2	7.2	87.20	49.20	15.53	50.64
SimpleRoute ($r = 4$)	99.2	58.4	12.0	**100.0**	98.2	43.6	41.8	20.0	12.2	80.33	58.87	22.60	53.93
SimpleRoute ($r = 8$)	99.2	54.6	18.0	**100.0**	**100.0**	67.0	59.6	53.6	23.0	86.27	69.40	36.00	63.89
SimpleRoute ($r = 16$)	99.8	**97.2**	**36.6**	**100.0**	99.6	41.0	93.2	43.6	6.4	97.67	80.13	28.00	68.60
SimpleRoute ($r = 64$)	**100.0**	71.4	25.4	**100.0**	**100.0**	**98.0**	67.6	85.6	**55.4**	89.20	85.67	**59.60**	78.16
SimpleRoute-Wide ($r = 8$)	99.6	68.6	25.6	**100.0**	99.8	93.2	**98.6**	**94.0**	52.4	**99.40**	**87.47**	57.07	**81.31**

C Complexity and efficiency of SimpleRoute

Router overhead. Routing adds only r write codes and r read codes, each a d_k-dimensional vector with a scalar bias that is matched against the per-head key or query of its assigned head, and token-local computation linear in sequence length and slot count. Specifically, the codes require $2r(d_k + 1)$ parameters per layer, their similarity computations cost $O(Lrd_k)$, and the two softmaxes cost $O(Lr)$ for a sequence of length L. For the 340M model ($d_k = 128$, 24 layers), this amounts to 24,768, 99,072, and 396,288 parameters for $r = 4$, 16, and 64, matching the differences from GLA in Table 2. All routing operations are token local and never form an $L \times L$ matrix.

Recurrent-state overhead. The recurrent-state overhead is exactly controlled by r/H and vanishes when $r = H$. The routed layer contains rd_kd_v state elements per layer, compared with Hd_kd_v for standard multi-head GLA, while recurrent work scales from $O(LHd_kd_v)$ to $O(Lrd_kd_v)$. Thus, $r = mH$ multiplies state size and recurrent work by m, and the decoding state remains independent of context length. Unlike state widening, SimpleRoute leaves the backbone projection and per-head state dimensions unchanged; $r > H$ only adds router-indexed states of the original shape.

Measured efficiency and length scaling. We evaluate whether SimpleRoute preserves the favorable length scaling of linear attention and quantify the overhead introduced by routing. We compare Transformer++, GLA, and SimpleRoute with $r \in \{4, 16\}$ from 2K to 16K context lengths, reporting training throughput, decoding FLOPs per generated token, and peak decoding memory in Figure 4. These measurements are efficiency benchmarks independent of the 2K training context. For training, we keep the number of tokens per optimization step fixed at 524,288 by reducing the micro-batch size as the context length increases. Decoding uses bf16 precision, batch size 1, and 256 generated tokens.

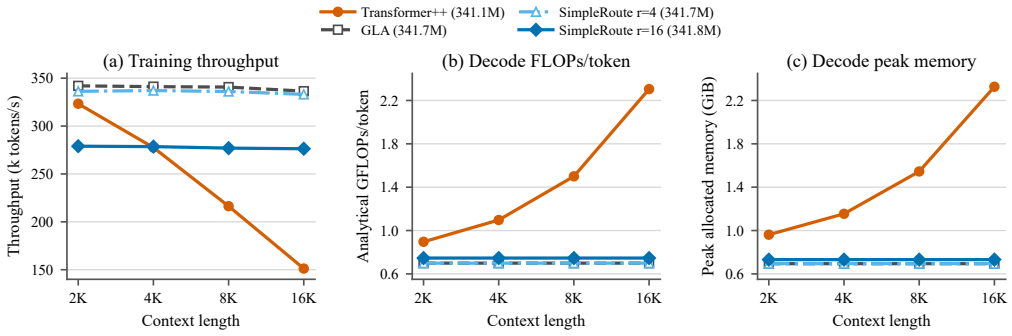

Figure 4: Computational scaling from 2K to 16K context lengths. (a) Measured training throughput with a fixed number of tokens per optimization step. (b) Analytical decoding FLOPs per generated token. (c) Peak allocated memory during decoding after the prompt.

SimpleRoute preserves the favorable length scaling of linear attention in both training and decoding. From 2K to 16K, Transformer++ retains only 46.8% of its 2K training throughput, whereas GLA and all SimpleRoute configurations retain 98.4–99.3%; at 16K, SimpleRoute with $r = 16$ is 1.83× faster than Transformer++. Over the same range, Transformer++ increases from 0.896 to 2.305 GFLOPs

per generated token and from 0.96 to 2.33 GiB of peak decoding memory, while the recurrent models remain nearly constant. At 16K, SimpleRoute reduces these two decoding costs by 67.6% and 68.6%, respectively. Thus, routing changes the constant computational cost but preserves linear training scaling and context-independent decoding cost.