TRIADIC LINEAR ATTENTION: THREE-DIMENSIONAL RECURRENT STATES FOR LONG-CONTEXT SEQUENCE MODELING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recurrent neural networks (RNNs) compress the historical context into a memory state of fixed size, thus allowing for constant-time inference. The memory state size is a crucial factor in their performance, as exemplified by the strong performance and resurgence of linear attention, which extends the vector-valued hidden states of ordinary RNNs to matrix-valued hidden states. Crucially, linear attention does so in a parameter-efficient way, in particular by using an outer product of the key and value vectors to write to the matrix-valued hidden state. We generalize this construction and propose *triadic linear attention*, which writes the triadic outer product of a key, a second key, and a value, into a third-order tensor state, and reads from it by contracting both key axes with two queries. An E-dimensional second key thus yields an E-fold increase in state size while adding only two projections. Triadic linear attention is compatible with data-dependent forgetting, the delta rule, and chunkwise-parallel training. Applied to Gated DeltaNet and scalar-gated linear attention, triadic linear attention substantially improves long-context language modeling and recall, outperforming alternatives that enlarge the state.

1 INTRODUCTION

Recent works on linear attention have highlighted the importance of *memory state size* in RNNs. Linear attention (Katharopoulos et al., 2020) extends vector-valued hidden states in ordinary RNNs to matrix-valued hidden states by using an outer product of two input-dependent vectors (i.e., the key and value vectors) to update the hidden state matrix. The increased state size has made it possible for modern RNNs to outperform classic RNNs despite making strong structural assumptions on the recurrence function—needed to parallelize the model across sequence length for efficient training.

State size determines how much of the context a model can recall (Arora et al., 2024)—a model with a fixed-size state cannot even copy sequences beyond a certain length (Jelassi et al., 2024). Modern linear attention models still struggle to retrieve information from their inputs and degrade on recall-intensive and long-context tasks (Arora et al., 2024; Hsieh et al., 2024). However, naïvely increasing the state size would increase parameter count significantly: adding more heads to the sequence mixer enlarges all its projections, and increasing the value dimension or putting multiple value heads per key still grows the value and output projections.

How can we increase the state size of linear attention in a parameter-efficient way? We observe that linear attention obtains its matrix-valued state by lifting the vector-valued state of classic RNNs by one order through a *dyadic* outer product of two vectors. This generalizes to an outer product of n vectors, yielding an n-way tensor state. This paper describes *triadic linear attention*, which uses $n = 3$ vectors (a key, a second key, and a value) and writes their triadic outer product into a third-order state. It reads from the state by contracting both key axes with two queries. This provides a parameter-efficient way to increase state size, since an E-dimensional second key and query give an *E-fold increase* in state size while adding only the two projections that produce them.

Triadic linear attention is compatible with the three main innovations of modern linear attention variants. First, data-dependent forgetting (Peng et al., 2021; Yang et al., 2023; Gu & Dao, 2024; Dao & Gu, 2024) extends naturally by giving each slice along the second-key axis its own forget gate. Second, the delta rule (Schlag et al., 2021; Yang et al., 2024; 2025b) extends by removing the value stored under both keys jointly before writing the new one. Third, triadic linear attention can

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.

be trained efficiently at scale with the chunkwise-parallel form (Hua et al., 2022; Sun et al., 2023; Yang et al., 2024), which we achieve by tiling the state along its value axis, so that no streaming multiprocessor (SM) ever holds the full third-order state of a head.

We apply triadic linear attention to Gated DeltaNet (GDN) and scalar-gated linear attention (sGLA), where we find that triadic linear attention substantially improves long-context language modeling and recall capabilities (see Figure 2). Our approach outperforms alternative approaches that also increase the state size through larger heads, larger values, more heads, or multiple value heads per key. We further demonstrate that a vanilla linear attention model can be converted to a triadic model post hoc by adapting a pretrained linear attention model to its triadic form during long-context extension. Finally, in a GDN/Transformer hybrid, making the GDN layers triadic lowers perplexity on long books by more than doubling the number of key-value heads does, while using significantly less memory at long context lengths. Together, our results suggest that triadic linear attention is a parameter-efficient way to enlarge state size and improve recurrent neural networks on long-context and recall-intensive tasks.

2 Triadic Linear Attention

2.1 Linear attention as an associative memory

Consider ordinary single-head linear attention on a sequence of token representations $\mathbf{x}_1, \ldots, \mathbf{x}_T$. Each token produces a query $\mathbf{q}_t$, a key $\mathbf{k}_t$ and a value $\mathbf{v}_t$ through learned linear projections of $\mathbf{x}_t$. At every step t, linear attention writes to a recurrent matrix state $\mathbf{S}_t$ through a rank-one update and reads from it by a matrix-vector product:

$$\mathbf{S}_t = \mathbf{S}_{t-1} + \mathbf{k}_t \mathbf{v}_t^\top, \qquad \mathbf{o}_t = \mathbf{S}_t^\top \mathbf{q}_t = \sum_{s \le t} \left(\mathbf{q}_t^\top \mathbf{k}_s\right) \mathbf{v}_s. \tag{1}$$

This formulation coincides with the fast-weight programmer of Schmidhuber (1992), where *slow-weights* emit representations to update a *fast-weight* network (Schlag et al., 2021). Here the fast weights (or equivalently, the hidden states of the RNN) are updated via a *dyadic* outer product of two input-dependent vectors. It is also a tensor product representation in the sense of Smolensky (1990), where a symbolic structure is embedded by binding each filler to its role through an outer product and storing all bindings in superposition by adding them up. Then, a filler is retrieved from this state by contracting it with a vector that matches the filler's role. In Equation 1 the value is the filler, the key is the role, $\mathbf{S}_t$ is the superposition, and the readout with the query is the unbinding. A lossless readout requires the query to be orthogonal to the keys of all other stored associations. Otherwise, the retrieval is lossy and every other value interferes in proportion to the overlap of its key and query. The appeal of this formulation is that storing associations beyond the capacity of the state degrades retrieval gradually, in contrast to other memory systems that evict entire entries.[1]

The capacity of the state is fully determined by its shape. With d-dimensional keys and values, $\mathbf{S}_t$ has d^2 entries and can separate at most d mutually orthogonal keys. Beyond this limit, interference is inevitable. Recent advances such as data-dependent forgetting (Yang et al., 2023; Dao & Gu, 2024) and the delta rule (Widrow et al., 1960; Schlag et al., 2021; Yang et al., 2024) improve *how* the state is managed, but every linear attention variant remains fundamentally bound by the same d^2-entry matrix. To lift this capacity limit, the state itself has to grow.

2.2 Binding to a second key

The tensor product view suggests a principled way to enlarge the state. As noted by Smolensky (1990), a binding itself is a vector (of dimension d^2 rather than d), and can therefore be bound to a further role. Each such nesting raises the order of the tensor memory by one. One application of this is the third-order tensor product representation of Schlag & Schmidhuber (2018), which stores a graph and learns its node and relation representations end-to-end. There, each edge is the outer product of its start node, relation and end node, the superposition of these encodes the graph, and unbinding with a start node and a relation retrieves the end node.

[1] Sliding-window attention is a modern instance of such an evicting memory system, since it keeps the last w key-value pairs exactly and removes previous representations entirely.

Here, we observe that the third order is useful beyond nested data, as a way to enlarge the capacity of the state. With orthonormal roles, an n-th order tensor product representation can store d^{n-1} associations exactly, so binding each value to a second key increases the capacity of linear attention from d to d^2 associations. The second key does not correspond to anything in the data and the model can freely choose how to use both keys to store associations. This can be seen as a general way of fast-weight programming a three-dimensional state.

We call the resulting model *triadic linear attention*, since the fast weights (i.e., hidden states) are obtained from a *triadic* outer product of three input-dependent vectors. Each position produces, besides $\mathbf{q}_t, \mathbf{k}_t, \mathbf{v}_t$ of dimension d, a second key $\mathbf{k}'_t$ and second query $\mathbf{q}'_t$ of dimension E, also through linear projections of $\mathbf{x}_t$. The state is a third-order tensor $\mathbf{S}_t$ of size $d \times E \times d$, fast-weight programmed through the outer product of three vectors and read by contracting its two key axes with two queries,

$$\mathbf{S}_t = \mathbf{S}_{t-1} + \mathbf{k}_t \otimes \mathbf{k}'_t \otimes \mathbf{v}_t, \qquad \mathbf{o}_t = \mathbf{S}_t \times_1 \mathbf{q}_t \times_2 \mathbf{q}'_t = \sum_{s \le t} \left(\mathbf{q}_t^\top \mathbf{k}_s\right)\left(\mathbf{q}_t'^\top \mathbf{k}'_s\right) \mathbf{v}_s, \tag{2}$$

where $\times_n$ denotes contraction along axis n. In the language of the previous subsection, the value is bound to two roles at once and unbound with two queries. For $E = 1$ with $\mathbf{k}'_t = \mathbf{q}'_t = 1$, Equation 2 reduces to Equation 1, so ordinary linear attention is a special case.

The state now has d^2E entries instead of d^2, which is cubic in d for $E = d$. Meanwhile, the parameters grow only by the two projections needed to produce $\mathbf{k}'_t$ and $\mathbf{q}'_t$, and therefore the ratio of state size to parameter count increases asymptotically. In practice, we keep E small (e.g., $E = 8$), which makes the parameter overhead negligible while multiplying the capacity of the state by E.

2.3 Capacity in isolation

Before adding further machinery, we verify that the construction of Equation 2 actually expands the capacity. To this end, we use multi-query associative recall (MQAR; Arora et al., 2023), where each example contains N key-value pairs followed by the same N keys in random order, and the model must predict the value corresponding to each key. We keep the model deliberately minimal, with two layers of four heads with $d = 16$ and a sequence mixer that is Equation 2, without convolutions, forgetting, non-linearities, or the delta rule. We compare plain linear attention (E=1) to triadic linear attention with $E \in \{2, 4, 8, 16\}$ across N from 32 to 4096. The detailed setup can be found in Appendix C. Figure 1 shows that every doubling of E shifts the accuracy curve to the right by roughly a doubling of N, e.g., for $E = 16$ the model stores roughly 16 times as many key-value associations as ordinary linear attention. Notably, this 16-fold increase in state capacity comes with only a 1.08-fold increase in non-embedding parameters.

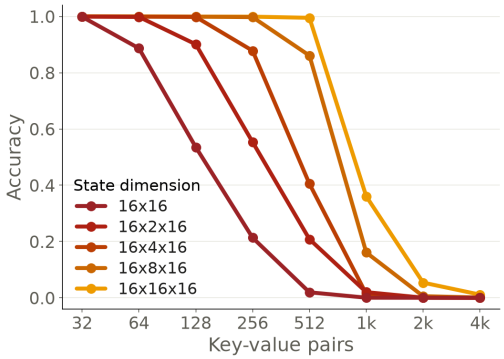

Figure 1: MQAR with plain linear attention (state 16×16) and triadic linear attention with $E \in \{2, 4, 8, 16\}$ (state 16×E×16).

2.4 Forgetting and the delta rule

Modern linear attention variants manage the state with data-dependent forgetting (Yang et al., 2023; Dao & Gu, 2024) and the delta rule (Widrow et al., 1960; Schlag et al., 2021; Yang et al., 2024), which removes (part of) the previously stored association before associating a key with a new value. We extend both ideas to the three-dimensional state.

Forgetting. Linear attention with scalar decay computes a gate $\alpha_t \in [0, 1]$ from x_t and multiplies the state with it before applying the rank-one update. In triadic linear attention, we let the decay vary along the second-key axis by giving each slice $\mathbf{S}_t[:, e, :]$ its own forget scalar $\alpha_{t,e}$. This yields the recurrence

$$\mathbf{S}_t = \mathbf{S}_{t-1} \times_2 \mathrm{diag}(\boldsymbol{\alpha}_t) + \mathbf{k}_t \otimes \mathbf{k}'_t \otimes \mathbf{v}_t. \tag{3}$$

Within a slice, the decay is scalar, and across slices it is channel-wise in the sense of gated linear attention (Yang et al., 2023). Since the second-key axis is small, the E different gates can be produced by a dense linear layer at negligible parameter cost.

Delta rule. Given a new key-value pair, the delta rule first removes what the state currently returns for the key, before writing the new association. For a three-dimensional state, the currently stored association is the two-key readout $\mathbf{S}_{t-1} \times_1 \mathbf{k}_t \times_2 \mathbf{k}'_t$. Removing exactly that value and interpolating toward the new one gives

$$\mathbf{S}_t = \mathbf{S}_{t-1} + \beta_t \, \mathbf{k}_t \otimes \mathbf{k}'_t \otimes \big(\mathbf{v}_t - \mathbf{S}_{t-1} \times_1 \mathbf{k}_t \times_2 \mathbf{k}'_t\big), \tag{4}$$

with a data-dependent write strength $\beta_t \in (0, 1)$. This can be combined with forgetting by decaying $\mathbf{S}_{t-1}$ before the erase, as in Equation 3.

Equation 4 applies the delta rule jointly to the pair of keys. By flattening the outer product of both keys into a joint key $\boldsymbol{\kappa}_t = \mathbf{k}_t \otimes \mathbf{k}'_t$ of dimension $d \cdot E$ and interpreting the state as a $d \cdot E \times d$ matrix, we can rewrite the update as

$$\mathbf{S}_t = \big(\mathbf{I} - \beta_t \, \boldsymbol{\kappa}_t \boldsymbol{\kappa}_t^\top\big)\mathbf{D}_t \, \mathbf{S}_{t-1} + \beta_t \, \boldsymbol{\kappa}_t \mathbf{v}_t^\top, \qquad \mathbf{o}_t = \mathbf{S}_t^\top \big(\mathbf{q}_t \otimes \mathbf{q}'_t\big), \tag{5}$$

where $\mathbf{D}_t$ is a diagonal decay matrix. This corresponds to Gated DeltaNet with a key dimension of $d \cdot E$, read with the $d \cdot E$-dimensional joint query $\mathbf{q}_t \otimes \mathbf{q}'_t$, and with a per-slice scalar decay.

2.5 Efficient implementation

Linear attention trains efficiently with the *chunkwise-parallel form* (Hua et al., 2022; Sun et al., 2023; Yang et al., 2024). The sequence is split into chunks of C positions, and $\square^r$ denotes the quantity $\square$ at position r of the current chunk. With the chunk's queries, keys and values stacked into the rows of $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{C \times d}$ and $\mathbf{S}_{[i]}$ the state at the start of the chunk, linear attention computes

$$\mathbf{O} = \mathbf{Q} \, \mathbf{S}_{[i]} + \mathrm{tril}\big(\mathbf{Q}\mathbf{K}^\top\big) \, \mathbf{V}, \qquad \mathbf{S}_{[i+1]} = \mathbf{S}_{[i]} + \mathbf{K}^\top \mathbf{V}, \tag{6}$$

so all positions of a chunk interact in parallel through masked attention, and only the state at chunk boundaries is carried by a recurrence.

Triadic linear attention is linear attention with the $d \cdot E$-dimensional joint key $\mathbf{k}_t \otimes \mathbf{k}'_t$, so Equation 6 applies in principle. Treating it this way, however, would cost E times more in every inner product of the masked attention, and E times more in the state, which no longer fits on chip. The Kronecker structure of the joint key removes the first cost, and tiling the state makes it fit on chip. We describe the chunkwise parallel form for triadic linear attention without forgetting or the delta rule here and defer the complete chunkwise parallel forms to Appendix E.

Separating the joint key. Inner products of joint queries and keys factorize, $(\mathbf{q}^r \otimes \mathbf{q}'^r)^\top (\mathbf{k}^s \otimes \mathbf{k}'^s) = (\mathbf{q}^{r\top} \mathbf{k}^s)(\mathbf{q}'^{r\top} \mathbf{k}'^s)$, so with $\mathbf{Q}', \mathbf{K}' \in \mathbb{R}^{C \times E}$ the masked attention of the chunk is

$$\big(\mathbf{Q}\mathbf{K}^\top\big) \odot \mathbf{R}', \qquad \mathbf{R}' = \mathrm{tril}\big(\mathbf{Q}'\mathbf{K}'^\top\big). \tag{7}$$

The interactions within a chunk reduce to ordinary linear attention with d-dimensional keys, whose causal mask is replaced by the $C \times C$ matrix $\mathbf{R}'$. The masked attention therefore works with $C \times C$ matrices, as in ordinary linear attention, and forming the matrix costs $C^2(d + E)$ operations instead of the $C^2(d \cdot E)$ of the joint keys. The only extra work over ordinary linear attention is the $C^2 E$ of computing $\mathbf{R}'$.

Tiling the state. The E-fold cost is thus confined to the state. Writing the $d \times E \times d$ state as slices $\mathbf{S}_{[i],e} \in \mathbb{R}^{d \times d}$ and $\mathbf{q}'_e, \mathbf{k}'_e \in \mathbb{R}^C$ for the columns of $\mathbf{Q}', \mathbf{K}'$, the chunk reads and writes each slice with the shared queries and keys,

$$\mathbf{O} = \sum_{e=1}^{E} \mathrm{diag}(\mathbf{q}'_e) \, \mathbf{Q} \, \mathbf{S}_{[i],e} + \big(\mathbf{Q}\mathbf{K}^\top \odot \mathbf{R}'\big) \, \mathbf{V}, \qquad \mathbf{S}_{[i+1],e} = \mathbf{S}_{[i],e} + \mathbf{K}^\top \mathrm{diag}(\mathbf{k}'_e) \, \mathbf{V}. \tag{8}$$

The slices interact only through the sum over e, and different columns of the value axis never interact. Our kernels therefore split the state of each head along the value axis into blocks of 32 columns, one per thread block, and each thread block keeps all E slices of its columns in registers for the entire sequence, so the full state is never held in one place. At $E = 8$, the state of one head occupies 512 KiB in FP32, twice the register file of a Hopper streaming multiprocessor, whereas one block of it occupies 128 KiB. Section 3.5 measures the resulting cost, and Appendix E describes the kernels.

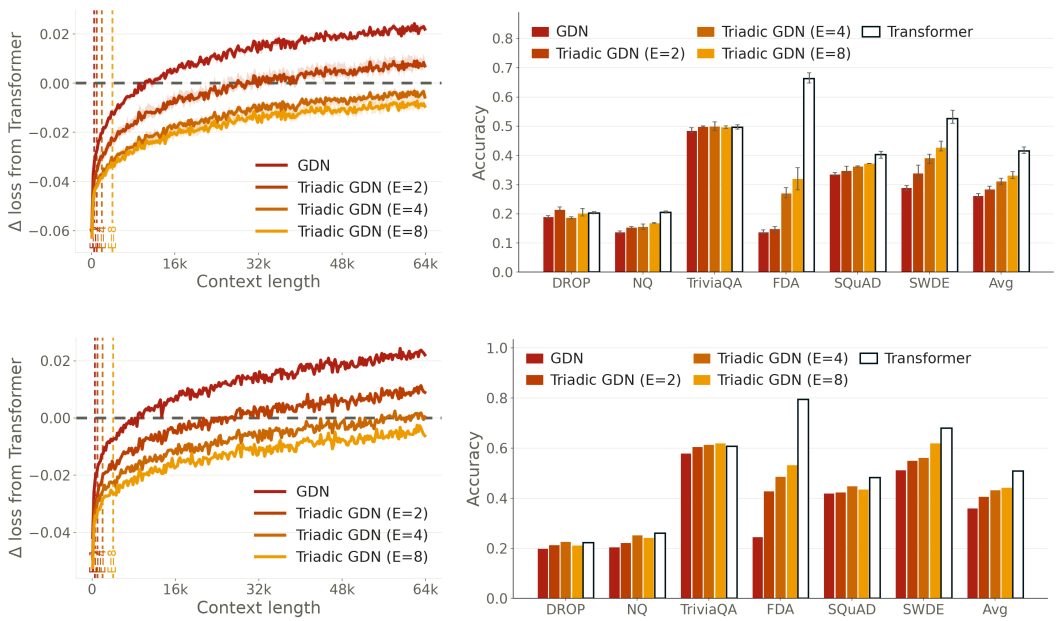

Figure 2: Triadic Gated DeltaNet with a growing second key dimension E at 400M parameters (top) and 1.3B parameters (bottom). Left: Loss difference to the Transformer on PG19 by context position, with dashed lines marking where the key-value cache of the Transformer exceeds the state of each model. Right: Accuracy on six recall-intensive tasks.

3 Empirical Study

3.1 Experimental Setup

Models. We train models at 400M parameters (24 layers, $d_{\text{model}} = 1024$, 8 heads) and 1.3B parameters (24 layers, $d_{\text{model}} = 2048$, 16 heads), with head dimension $d = 128$ throughout. We apply triadic linear attention to two base sequence mixers, Gated DeltaNet (GDN; Yang et al., 2025b) and scalar-gated linear attention (sGLA), which we define as GDN without the erase term of the delta rule.[2] The triadic variants keep the base mixer and add a second key and query, which are obtained through a linear projection followed by a short convolution and a softplus activation. Additionally, each slice obtains a separate scalar forget gate, as described in Section 2.4. The detailed architecture is described in Appendix D. For $E = 1$, Triadic GDN reduces exactly to Gated DeltaNet, while $E = 8$ adds only 1.2% more parameters. We train $E \in \{1, 2, 4, 8\}$ for both variants at 400M and for Triadic GDN at 1.3B.

Training. We pretrain each model on 50 tokens per parameter, i.e., 2.5× Chinchilla-optimal (Hoffmann et al., 2022), which corresponds to 20B tokens for the 400M parameter models and 65B tokens for the 1.3B parameter models. We train on Fineweb-Edu (Lozhkov et al., 2024) with context length 4k, then long-context extend all models to 64k on 5 tokens per parameter using a mixture of Fineweb-Edu, PG19 books data (Rae et al., 2020) and scientific PDFs (Olmo et al., 2026). We use the AdamW optimizer (Loshchilov & Hutter, 2019) with peak learning rate 3×10^{-4} for pretraining and peak learning rate 10^{-4} for long context extension, with weight decay 0.1 throughout. The learning rate is warmed up linearly and then decays with a cosine schedule to 10% of its peak. For both training stages, we use an effective batch size of about 0.5M tokens at 400M parameters and about 1M tokens at 1.3B parameters.

[2] sGLA thereby belongs to the same family as other linear attention models with a data-dependent scalar decay, such as gated RFA (Peng et al., 2021), Mamba-2 (Dao & Gu, 2024) and mLSTM (Beck et al., 2024). However, Mamba-2 and xLSTM have other components which we do not make use of, and hence we use sGLA to describe linear attention with scalar data-dependent gates.

Table 1: Different ways of enlarging the state size of Gated DeltaNet (GDN) and scalar-gated linear attention (sGLA) by $2\times$ and $4\times$ at matched parameter count. State in MB (assuming 2 bytes per entry), and parameters in millions. Results are averaged over three training seeds.

Model	State	Params	0-shot ↑	Wiki. ↓	PG19 ≤4k ↓	PG19 4k–16k ↓	PG19 16k–64k ↓	Recall ↑
Transformer	12.6/1k tok	377.0	52.6	11.11	15.39	14.41	13.92	41.6
GDN (base)	6.3	380.9	53.1	11.25	15.03	14.38	14.15	26.2
Larger heads (d=256)	12.6	380.7	52.6	11.22	15.10	14.42	14.17	26.6
Grouped values (2 per key)	12.6	378.2	52.9	11.18	15.05	14.37	14.12	27.1
Wider values (d_v=256)	12.6	377.9	53.2	11.20	15.07	14.39	14.14	27.4
More heads (16)	12.6	381.6	53.3	11.32	15.25	14.54	14.28	27.5
Triadic (E=2)	12.6	381.9	**53.4**	**11.08**	**14.90**	**14.21**	**13.95**	**28.4**
Larger heads (d=512)	25.2	380.6	53.0	11.25	15.21	14.50	14.24	28.5
Grouped values (4 per key)	25.2	382.4	52.7	11.35	15.35	14.63	14.36	27.3
Wider values (d_v=512)	25.2	381.3	52.5	11.41	15.41	14.70	14.44	27.5
Triadic (E=4)	25.2	383.0	**53.6**	**10.93**	**14.80**	**14.08**	**13.79**	**31.1**
sGLA (base)	6.3	380.9	53.5	11.41	15.19	14.55	14.32	26.4
Larger heads (d=256)	12.6	380.7	52.4	11.38	15.22	14.57	14.33	28.2
Grouped values (2 per key)	12.6	378.2	53.0	11.34	15.19	14.53	14.28	**28.5**
Wider values (d_v=256)	12.6	377.9	53.5	11.35	15.19	14.53	14.28	27.7
More heads (16)	12.6	381.6	52.9	11.42	15.36	14.67	14.42	27.8
Triadic (E=2)	12.6	381.9	**53.5**	**11.19**	**15.00**	**14.32**	**14.07**	28.4
Larger heads (d=512)	25.2	380.6	52.2	11.52	15.53	14.85	14.59	28.7
Grouped values (4 per key)	25.2	382.4	52.8	11.47	15.50	14.79	14.52	28.3
Wider values (d_v=512)	25.2	381.3	52.4	11.51	15.52	14.82	14.55	29.0
Triadic (E=4)	25.2	383.0	**53.4**	**11.09**	**14.94**	**14.24**	**13.97**	**30.7**

Baselines. In addition to ordinary sGLA/GDN baselines, we compare against established methods for increasing the state size, compared at matched state size. Specifically, we train four variants of each model with $2\times$ and $4\times$ its state through a larger head dimension with fewer heads (*larger heads* in Table 1), a larger value dimension (*wider values*), several value heads per key head (*grouped values*), or more heads (*more heads*). A larger head dimension leaves the total width of projections unchanged, while the other variants enlarge the linear projections of the sequence mixer significantly. We account for this by reducing the MLP width to match the parameter count. Adding heads is only feasible at $2\times$, since at $4\times$ the projections of the sequence mixer alone exceed the parameter budget. For reference, we also train Transformers with RoPE (Su et al., 2021), QK-norm (Dehghani et al., 2023) and grouped-query attention (Ainslie et al., 2023) with eight query heads per key-value head. We use a RoPE base frequency of 10k during pretraining and raise it to 2M for long-context extension.

Evaluation. We evaluate all models after long-context extension. We report perplexity on held-out PG19 books (Rae et al., 2020) by context position and perplexity on Wikitext (Merity et al., 2016). To measure in-context recall, we use the benchmark suite from Arora et al. (2024), and additionally report the average over ten zero-shot tasks. More details are given in Appendix D.

3.2 Main Results

Scaling the state. Figure 2 shows how the state size of triadic linear attention affects language modeling and in-context recall at the 400M and 1.3B parameter scales. Gated DeltaNet predicts early tokens better than the Transformer but falls behind at a context length of around 10k tokens. With $E = 2$, triadic Gated DeltaNet pushes this crossover point considerably further out, and with $E = 8$, it predicts the next token better than the Transformer even at 64k context. The dashed vertical lines indicate the context lengths at which the key-value cache of the Transformer exceeds the state size of the recurrent model, so beyond this point, triadic Gated DeltaNet predicts the next token better with a smaller state. Overall, we observe language modeling to improve as the state size grows, but with diminishing returns. On recall-intensive tasks, triadic Gated DeltaNet substantially improves performance at both scales, with particularly large gains on FDA and SWDE, which require copying information from a long document. Figure 3 shows the same trend on the needle-in-a-haystack tasks of RULER (Hsieh et al., 2024), where a larger state keeps retrieval accurate up to longer contexts. Appendix F shows that the larger state is indeed used for distant context, as Triadic GDN with $E = 8$ benefits more than GDN from additional context when predicting the same final 4k tokens.

Table 2: Upcycling a pretrained model from $E = 1$ to $E = 8$ during long-context extension, compared with the corresponding base model and with $E = 8$ pretrained from scratch. Results are averaged over three seeds at 400M and reported for a single seed at 1.3B.

Model	State	Params	0-shot ↑	Wiki. ↓	PG19 ≤4k ↓	PG19 4k–16k ↓	PG19 16k–64k ↓	Recall ↑
GDN (base)	6.3	380.9	53.1	11.25	15.03	14.38	14.15	26.2
Triadic (E=8), from scratch	50.3	385.4	**53.4**	**10.87**	**14.79**	**14.04**	**13.74**	**33.1**
Triadic (E=8), upcycled	50.3	385.4	53.2	11.04	14.91	14.18	13.86	29.7
sGLA (base)	6.3	380.9	**53.5**	11.41	15.19	14.55	14.32	26.4
Triadic (E=8), from scratch	50.3	385.4	53.4	**11.03**	**14.96**	**14.23**	**13.95**	**32.1**
Triadic (E=8), upcycled	50.3	385.4	53.3	11.16	15.04	14.32	14.03	30.0
GDN (1.3B, base)	12.6	1360.2	59.6	8.08	10.91	10.38	10.20	36.1
Triadic (E=8), from scratch	100.7	1378.3	**60.9**	**7.87**	**10.74**	**10.16**	**9.94**	**44.4**
Triadic (E=8), upcycled	100.7	1378.3	59.8	7.96	10.82	10.24	10.00	42.0
sGLA (1.3B, base)	12.6	1359.4	60.3	8.26	11.10	10.61	10.45	37.0
Triadic (E=8), from scratch	100.7	1377.5	**60.4**	**7.94**	**10.82**	**10.25**	**10.03**	**45.1**
Triadic (E=8), upcycled	100.7	1377.5	60.4	8.12	10.97	10.40	10.19	42.2

State-matched comparison. Table 1 compares triadic linear attention to alternative methods for increasing the state size. At both $2\times$ and $4\times$, triadic Gated DeltaNet achieves consistently lower perplexity on WikiText and on every PG19 range, while scoring the highest average on the recall evaluation suite. The alternative methods barely improve upon the vanilla Gated DeltaNet at $2\times$, and at $4\times$ they even degrade on every PG19 range. There, two of the alternatives (larger values, more value heads) require so many additional parameters that the MLP width has to be reduced from 2816 to 640 to stay within the parameter budget. Triadic linear attention instead adds only two small projections, which keeps the parameter overhead low and preserves the full MLP. The comparison for sGLA is similar, with triadic sGLA achieving the lowest perplexity at both state sizes.

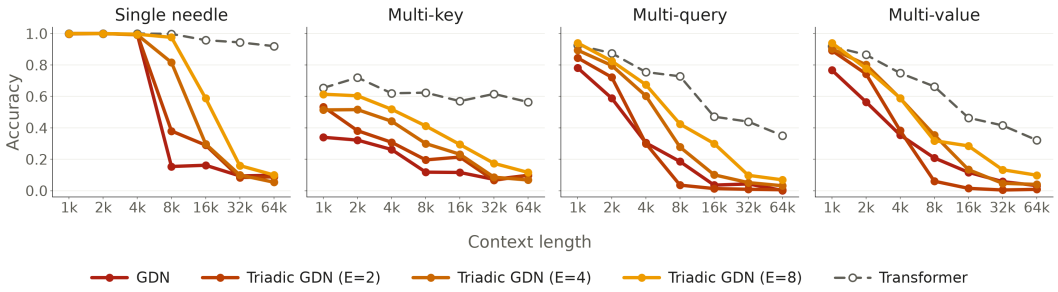

Figure 3: RULER needle-in-a-haystack accuracy by context length at 1.3B parameters.

3.3 Upcycling to Triadic Linear Attention via Continued Pretraining

We also test whether an ordinary linear attention model can be made triadic after pretraining. To this end, we expand a pretrained model from $E = 1$ to $E = 8$ by copying its forget gates to every slice and initializing the second key and query projections as well as their convolutions from scratch. Then, we perform the long-context extension on the resulting triadic model. Table 2 shows that the upcycled models consistently improve over their base models for both sequence mixers and at both scales. They recover about half to three quarters of the gain that a triadic model pretrained from scratch achieves over its dyadic counterpart. Enlarging the state when training on longer context lengths is reminiscent of how the key-value cache of softmax attention grows with the context. Since short-context data requires little state but makes up most of the pretraining data, a linear attention model could potentially be pretrained with a small state that is enlarged in later training stages with higher sequence lengths.

3.4 Hybrid Models

In practice, linear attention variants are often deployed interleaved with softmax attention blocks (Team et al., 2025; Yang et al., 2025a). This raises a natural question: Is it more beneficial to enlarge the state of linear attention blocks or the key-value cache of the softmax attention blocks?

Table 3: Enlarging either the key-value cache (GQA-4) or the linear-attention state (Triadic, $E = 4$) in a 3:1 GDN/GQA-8 hybrid at 400M parameters. State is reported in MB at 4k and 64k tokens. NIAH is averaged over eight RULER tasks. Results are averaged over three seeds.

Model	State @ 4k ↓	State @ 64k ↓	0-shot ↑	Wiki. ↓	PG19 ≤4k ↓	PG19 4k–64k ↓	Recall ↑	NIAH Avg. ↑
Transformer	50	805	52.6	11.11	15.39	14.02	41.6	48.2
GDN	6	6	53.1	11.25	15.03	14.19	26.2	21.7
3:1 GDN/GQA-8 (base)	17	206	53.0	10.69	14.86	13.50	43.7	51.5
Larger KV cache (GQA-4)	30	407	53.5	10.67	14.83	13.47	**45.3**	53.0
Triadic state (E=4)	31	220	**53.8**	**10.63**	**14.76**	**13.39**	44.5	**56.3**

To test this, we train a 3:1 GDN/GQA-8 hybrid with NoPE (Kazemnejad et al., 2023), where GQA-g denotes grouped-query attention with g query heads per key-value head, and compare it to the same model with twice as large key-value cache (3:1 GDN/GQA-4) and to the same model with four times larger linear attention states (3:1 Triadic GDN/GQA-8 (E=4)). Table 3 shows that the triadic hybrid achieves the lowest perplexity at every context range as well as the best zero-shot and NIAH averages, and trails 3:1 GDN/GQA-4 only on recall (within seed spread). At the same time, 3:1 GDN/GQA-4 requires more memory than the triadic hybrid beyond about 4.6k tokens and almost twice as much at 64k tokens.

3.5 Efficiency

Figure 4 compares the forward and backward time of one block per sequence of the 1.3B models from 2k to 64k tokens at batch size 4, measured on H100 GPUs. Triadic GDN uses our CuTe kernels (Section 2.5 and Appendix E), GDN uses the TileLang kernels of FlashQLA (Zhang et al., 2026a), and the Transformer uses FlexAttention (Dong et al., 2024). GDN matches the Transformer at 2k tokens, and Triadic GDN overtakes it at 4k for $E = 2$ and $E = 4$ and at 8k for $E = 8$. With $E = 8$, it is 3% slower than the Transformer at 4k and faster at every longer context, reaching 5.1 times faster at 64k. In general, the overhead of Triadic GDN over vanilla Gated DeltaNet is 28%–30% for $E = 8$, 14%–15% for $E = 4$, and 9%–11% for $E = 2$.

4 Discussion and Limitations

Our results indicate that state size is an important axis for improving linear RNNs, in particular on recall-intensive tasks and at long context. Triadic linear attention provides a parameter-efficient way to increase it, similar to how (dyadic) linear attention increases the state size of traditional vector-valued RNNs. At both 400M and 1.3B scales, triadic linear attention consistently improves perplexity and recall, and it outperforms alternative methods to enlarge the state size.

Nevertheless, the larger state slows down training by around 15% for $E = 4$ and by around 30% for $E = 8$, even with our optimized kernels. Additionally, performance on some recall-intensive tasks still trails that of Transformers, but only with a far larger state size at long context. Finally, we apply the triadic construction only to GDN and sGLA, which is only a small subset of the linear attention literature. Many other developments for matrix-valued states could be revisited for three-dimensional states, and the additional axis opens up possibilities for entirely new variants.

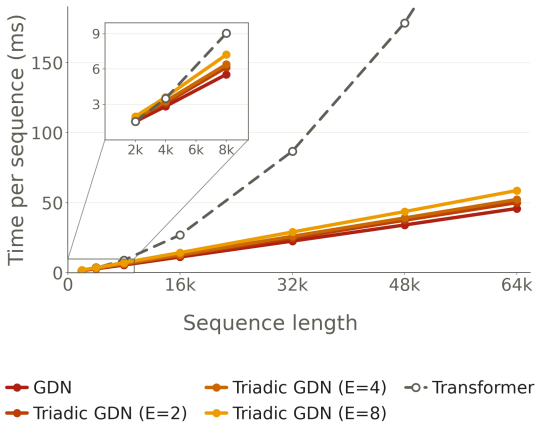

Figure 4: Forward and backward time of one block of the 1.3B models per sequence, at batch size 4 on H100 GPUs. Triadic GDN with $E = 8$ adds moderate overhead over GDN and is faster than the Transformer beyond 4k tokens.

5 Related Work

Linear attention. Linear attention replaces the softmax over the attention logits with a kernel feature map. This allows the growing key-value cache to collapse into a fixed-size recurrent state. The resulting recurrence writes to a matrix state by an outer product of a key and a value and reads from it using the query (Katharopoulos et al., 2020). Up to the normalization factor omitted in modern variants, this update coincides with the fast weight programmer of Schmidhuber (1992), as observed by Schlag et al. (2021). Early work on linear attention focused on better feature maps (Peng et al., 2021; Choromanski et al., 2021; Arora et al., 2024), a line of work directly relevant here, since our three-dimensional state can be interpreted as an ordinary matrix state under a feature map that expands the concatenated key $[k, k'] \in \mathbb{R}^{d+E}$ into the $d \cdot E$-dimensional key $k \otimes k'$. Subsequent work introduced forgetting to the recurrence, first through fixed decay (Sun et al., 2023) and later through data-dependent gates (Yang et al., 2023; Dao & Gu, 2024; Beck et al., 2024; Qin et al., 2024; Peng et al., 2024). A complementary development adds the delta rule (Widrow et al., 1960) into linear attention, yielding DeltaNet (Schlag et al., 2021; Yang et al., 2024). Combining this with data-dependent forgetting gives Gated DeltaNet (Yang et al., 2025b). More recent work generalizes the Gated DeltaNet recurrence to identity-plus-low-rank state transitions and adds more flexible parametrizations (Peng et al., 2025; Hatamizadeh et al., 2026).

Increasing state size. Arora et al. (2024) identify a tradeoff between recurrent state size and recall, and we adopt their recall-focused benchmark suite. Dense linear-attention variants can increase state size through larger key or value dimensions, more heads, or multiple value heads per key head (Gu & Dao, 2024; Dao & Gu, 2024; Yang et al., 2025a). These changes stay close to the original formulation and therefore serve as our main baselines in Section 3. A second line of work uses a fixed set of memory slots and routes over them (Peng et al., 2022; Zhang et al., 2024; Afzal et al., 2026). Mixture-of-Memories extends this idea to multiple matrix-valued states with sparse routing (Du et al., 2026). Related methods use product-key addressing (Lample et al., 2019; Zhao & Jones, 2026; Cabannes et al., 2026) to manage very large memories sparsely. These approaches generally have much higher state size compared to our construction, which comes with substantial computational overhead for memory movement. Triadic linear attention instead uses a single dense state and admits efficient fused kernels, which makes the increase in state size practical.

Higher-order associative memories. The matrix state of linear attention stores associations as a sum of outer products and is thus a correlation matrix memory (Kohonen, 1972). In that literature, moving from pairwise to higher-order interactions is a standard route to higher capacity (Poggio, 1975; Chen et al., 1986). The number of storable patterns then scales as d^{n-1}, where d is the number of memory units and n is the interaction order (Baldi & Venkatesh, 1987). This line of work was revived as dense associative memory (Krotov & Hopfield, 2016) and later related to attention (Ramsauer et al., 2021). Smolensky (1990) interpreted the same construction from a symbolic perspective, where a filler is bound to a role by an outer product and unbound by contraction. The closest prior work to ours is Schlag & Schmidhuber (2018), who place a third-order state inside a recurrent network. As in our work, each write updates a third-order tensor with a tensor product, but their recurrent update also includes three operations specifically designed for graph traversal (write, move, backlink). Their work focuses on compositional reasoning on small-scale language tasks, while we use a similar construction to enlarge the state of a modern linear-attention layer. The resulting sequence mixer incorporates a multi-head architecture, short convolutions, nonlinearities, normalizations, data-dependent forgetting, and a learnable delta rule, while efficient kernels allow us to train stacks of these layers at scale for general language modeling.

6 Conclusion

We introduced triadic linear attention, which updates a third-order tensor state with the outer product of three vectors and thereby increases the capacity of constant-time sequence mixers. Compared to other methods, this construction adds little parameter overhead and admits efficient kernels that exploit its structure. Across language modeling experiments at the 400M and 1.3B parameter scales, triadic Gated DeltaNet consistently lowers perplexity and improves recall, while custom kernels keep the training time within 1.3 times that of vanilla Gated DeltaNet for $E = 8$. We hope that triadic linear attention is a starting point for linear attention variants with larger states and ultimately for pure linear attention models at the frontier.

AI USE STATEMENT

In this work, we used generative AI tools for parts of the implementation. We have not used generative AI tools for generating synthetic datasets, developing theoretical or conceptual models, proposing or refining hypotheses, designing the research methodology or experiments, translation, cleaning or reformatting datasets, qualitative or thematic analysis, or interpreting results. Formulating mathematical claims and writing proofs are not applicable to this work. We have verified all AI-assisted implementations carefully and take responsibility for their correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Arshia Afzal, Aviv Bick, Eric P. Xing, Volkan Cevher, and Albert Gu. Raven: High-recall sequence modeling with sparse memory routing. 2026.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Zou, Atri Rudra, and Christopher Ré. Zoology: Measuring and improving recall in efficient language models. *arXiv preprint arXiv:2312.04927*, 2023.

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, Dylan Zinsley, James Zou, Atri Rudra, and Christopher Ré. Simple linear attention language models balance the recall-throughput tradeoff. In *Proceedings of ICML*, 2024.

Pierre Baldi and Santosh S. Venkatesh. Number of stable points for spin-glasses and neural networks of higher orders. *Physical Review Letters*, 58(9):913–916, 1987. doi: 10.1103/PhysRevLett.58.913.

Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael K Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xLSTM: Extended long short-term memory. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=ARAxPPIAhq.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-3786. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/a4ca07aa108036f80cbb5b82285fd4b1-Abstract-Conference.html.

Ali Behrouz, Zeman Li, Praneeth Kacham, Majid Daliri, Yuan Deng, Peilin Zhong, Meisam Razaviyayn, and Vahab Mirrokni. ATLAS: Learning to optimally memorize the context at test time. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2505.23735.

Yonatan Bisk, Rowan Zellers, Ronan LeBras, Jianfeng Gao, and Yejin Choi. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 7432–7439. AAAI Press, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/6239.

Loïc Cabannes, Pierre-Emmanuel Mazaré, Gergely Szilvasy, Matthijs Douze, Maria Lomeli, Ilze Amanda Auzina, Justin Carpentier, Gabriel Synnaeve, and Hervé Jégou. Sparse delta memory: Scaling the state of linear rnns through sparsity, 2026. URL https://arxiv.org/abs/2607.07386.

H. H. Chen, Y. C. Lee, G. Z. Sun, H. Y. Lee, T. Maxwell, and C. L. Giles. High order correlation model for associative memory. In John S. Denker (ed.), *Neural Networks for Computing*, volume 151 of *AIP Conference Proceedings*, pp. 86–99. American Institute of Physics, 1986. doi: 10.1063/1.36224.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *NAACL*, 2019.

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 preprint*, abs/1803.05457, 2018. URL https://arxiv.org/abs/1803.05457.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 10041–10071. PMLR, 2024.

Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Peter Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, Rodolphe Jenatton, Lucas Beyer, Michael Tschannen, Anurag Arnab, Xiao Wang, Carlos Riquelme Ruiz, Matthias Minderer, Joan Puigcerver, Utku Evci, Manoj Kumar, Sjoerd Van Steenkiste, Gamaleldin Fathy Elsayed, Aravindh Mahendran, Fisher Yu, Avital Oliver, Fantine Huot, Jasmijn Bastings, Mark Collier, Alexey A. Gritsenko, Vighnesh Birodkar, Cristina Nader Vasconcelos, Yi Tay, Thomas Mensink, Alexander Kolesnikov, Filip Pavetic, Dustin Tran, Thomas Kipf, Mario Lucic, Xiaohua Zhai, Daniel Keysers, Jeremiah J. Harmsen, and Neil Houlsby. Scaling vision transformers to 22 billion parameters. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 7480–7512. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/dehghani23a.html.

Juechu Dong, Boyuan Feng, Driss Guessous, Yanbo Liang, and Horace He. Flex attention: A programming model for generating optimized attention kernels, 2024. URL https://arxiv.org/abs/2412.05496.

Jusen Du, Weigao Sun, Disen Lan, Jiaxi Hu, Tao Zhang, and Yu Cheng. Mom: Linear sequence modeling with mixture-of-memories. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=3PdOq8Rgue.

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. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *Proceedings of CoLM*, 2024.

Ali Hatamizadeh, Yejin Choi, and Jan Kautz. Gated deltanet-2: Decoupling erase and write in linear attention, 2026. URL https://arxiv.org/abs/2605.22791.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, DDL Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 10, 2022.

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

Weizhe Hua, Zihang Dai, Hanxiao Liu, and Quoc Le. Transformer quality in linear time. In *International conference on machine learning*, pp. 9099–9117. PMLR, 2022.

Kazuki Irie, Róbert Csordás, and Jürgen Schmidhuber. The dual form of neural networks revisited: Connecting test time predictions to training patterns via spotlights of attention. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9639–9659. PMLR, 2022. URL https://proceedings.mlr.press/v162/irie22a.html.

Samy Jelassi, David Brandfonbrener, Sham M. Kakade, and Eran Malach. Repeat after me: Transformers are better than state space models at copying. 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. 21502–21521. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/jelassi24a.html.

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

Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. The impact of positional encoding on length generalization in transformers. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Teuvo Kohonen. Correlation matrix memories. *IEEE Transactions on Computers*, C-21(4):353–359, 1972. doi: 10.1109/TC.1972.5008975.

Dmitry Krotov and John J. Hopfield. Dense associative memory for pattern recognition. In *Advances in Neural Information Processing Systems*, volume 29, pp. 1172–1180, 2016.

Guillaume Lample, Alexandre Sablayrolles, Marc'Aurelio Ranzato, Ludovic Denoyer, and Herve Jegou. *Large memory layers with product keys*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. Fineweb-edu: the finest collection of educational content, 2024. URL https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391. Association for Computational Linguistics, 2018. doi: 10.18653/v1/D18-1260.

Team Olmo, :, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, Pradeep Dasigi, Robert Berry, Saumya Malik, Saurabh Shah, Scott Geng, Shane Arora, Shashank Gupta, Taira Anderson, Teng Xiao, Tyler Murray, Tyler Romero, Victoria Graf, Akari Asai, Akshita Bhagia, Alexander Wettig, Alisa Liu, Aman Rangapur, Chloe Anastasiades, Costa Huang, Dustin Schwenk, Harsh Trivedi, Ian Magnusson, Jaron Lochner, Jiacheng Liu, Lester James V. Miranda, Maarten Sap, Malia Morgan, Michael Schmitz, Michal

Guerquin, Michael Wilson, Regan Huff, Ronan Le Bras, Rui Xin, Rulin Shao, Sam Skjonsberg, Shannon Zejiang Shen, Shuyue Stella Li, Tucker Wilde, Valentina Pyatkin, Will Merrill, Yapei Chang, Yuling Gu, Zhiyuan Zeng, Ashish Sabharwal, Luke Zettlemoyer, Pang Wei Koh, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. Olmo 3, 2026. URL https://arxiv.org/abs/2512.13961.

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 Katrin Erk and Noah A. Smith (eds.), *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1525–1534, Berlin, Germany, 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1144. URL https://aclanthology.org/P16-1144.

Bo Peng, Daniel Goldstein, Quentin Anthony, Alon Albalak, Eric Alcaide, Stella Biderman, Eugene Cheah, Teddy Ferdinan, Haowen Hou, Przemysław Kazienko, et al. Eagle and finch: Rwkv with matrix-valued states and dynamic recurrence. *arXiv preprint arXiv:2404.05892*, 3, 2024.

Bo Peng, Ruichong Zhang, Daniel Goldstein, Eric Alcaide, Xingjian Du, Haowen Hou, Jiaju Lin, Jiaxing Liu, Janna Lu, William Merrill, et al. Rwkv-7" goose" with expressive dynamic state evolution. *arXiv preprint arXiv:2503.14456*, 2025.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah Smith, and Lingpeng Kong. Random feature attention. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=QtTKTdVrFBB.

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 Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7469–7483, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.515. URL https://aclanthology.org/2022.acl-long.515/.

Tomaso Poggio. On optimal nonlinear associative recall. *Biological Cybernetics*, 19(4):201–209, September 1975. doi: 10.1007/BF02281970.

Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. HGRN2: Gated Linear RNNs with State Expansion. In *Proceedings of CoLM*, 2024.

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. In *Proceedings of ICLR*, 2020.

Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Thomas Adler, Lukas Gruber, Markus Holzleitner, David Kreil, Michael K. Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. Hopfield networks is all you need. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=tL89RnzIiCd.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S. Gordon. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *Logical Formalizations of Commonsense Reasoning, Papers from the 2011 AAAI Spring Symposium (Technical Report SS-11-06)*, pp. 90–95. AAAI Press, 2011.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 8732–8740. AAAI Press, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/6399.

Imanol Schlag and Jürgen Schmidhuber. Learning to reason with third-order tensor products. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 10003–10014, Red Hook, NY, USA, 2018. Curran Associates Inc.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear Transformers Are Secretly Fast Weight Programmers. In *Proceedings of ICML*, 2021.

Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992.

Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

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

Julien Siems, Timur Carstensen, Arber Zela, Frank Hutter, Massimiliano Pontil, and Riccardo Grazzi. Deltaproduct: Improving state-tracking in linear rnns via householder products. *arXiv preprint arXiv:2502.10297*, 2025.

P. Smolensky. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artif. Intell.*, 46(1–2):159–216, November 1990. ISSN 0004-3702. doi: 10.1016/0004-3702(90)90007-M. URL https://doi.org/10.1016/0004-3702(90)90007-M.

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

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): Rnns with expressive hidden states, 2025. URL https://arxiv.org/abs/2407.04620.

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.

Arnuv Tandon, Karan Dalal, Xinhao Li, Daniel Koceja, Marcel Rød, Sam Buchanan, Xiaolong Wang, Jure Leskovec, Sanmi Koyejo, Tatsunori Hashimoto, Carlos Guestrin, Jed McCaleb, Yejin Choi, and Yu Sun. End-to-end test-time training for long context, 2025. URL https://arxiv.org/abs/2512.23675.

Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, Wentao Li, Enzhe Lu, Weizhou Liu, Yanru Chen, Weixin Xu, Longhui Yu, Yejie Wang, Yu Fan, Longguang Zhong, Enming Yuan, Dehao Zhang, Yizhi Zhang, T. Y. Liu, Haiming Wang, Shengjun Fang, Weiran He, Shaowei Liu, Yiwei Li, Jianlin Su, Jiezhong Qiu, Bo Pang, Junjie Yan, Zhejun Jiang, Weixiao Huang, Bohong Yin, Jiacheng You, Chu Wei, Zhengtao Wang, Chao Hong, Yutian Chen, Guanduo Chen, Yucheng Wang, Huabin Zheng, Feng Wang, Yibo Liu, Mengnan Dong, Zheng Zhang, Siyuan Pan, Wenhao Wu, Yuhao Wu, Longyu Guan, Jiawen Tao, Guohong Fu, Xinran Xu, Yuzhi Wang, Guokun Lai, Yuxin Wu, Xinyu Zhou, Zhilin Yang, and Yulun Du. Kimi linear: An expressive, efficient attention architecture, 2025. URL https://arxiv.org/abs/2510.26692.

Vijay Thakkar, Pradeep Ramani, Cris Cecka, Aniket Shivam, Honghao Lu, Ethan Yan, Jack Kosaian, Mark Hoemmen, Haicheng Wu, Andrew Kerr, Matt Nicely, Duane Merrill, Dustyn Blasig, Aditya Atluri, Fengqi Qiao, Piotr Majcher, Paul Springer, Markus Hohnerbach, Jin Wang, and Manish Gupta. CUTLASS, January 2023. URL https://github.com/NVIDIA/cutlass.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi,

Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

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

Johannes von Oswald, Nino Scherrer, Seijin Kobayashi, Luca Versari, Songlin Yang, Maximilian Schlegel, Kaitlin Maile, Yanick Schimpf, Oliver Sieberling, Alexander Meulemans, et al. Mesanet: Sequence modeling by locally optimal test-time training. *arXiv preprint arXiv:2506.05233*, 2025.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pp. 94–106. Association for Computational Linguistics, 2017. doi: 10.18653/v1/W17-4413.

Bernard Widrow, Marcian E Hoff, et al. Adaptive switching circuits. In *IRE WESCON convention record*, volume 4, pp. 96–104. New York, 1960.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025a. URL https://arxiv.org/abs/2505.09388.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2023.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In *Proceedings of NeurIPS*, 2024.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *Proceedings of ICLR*, 2025b.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472.

Chengruidong Zhang, Xi Lin, Huiqiang Jiang, Zekun Wang, Xiao Li, Yizhong Cao, Bohan Zhuang, Rui Men, Jianwei Zhang, Bo Zheng, Junyang Lin, Dayiheng Liu, and Jingren Zhou. Flashqla: Flash qwen linear attention. https://github.com/QwenLM/FlashQLA, 2026a.

Tianyuan Zhang, Sai Bi, Yicong Hong, Kai Zhang, Fujun Luan, Songlin Yang, Kalyan Sunkavalli, William T. Freeman, and Hao Tan. Test-time training done right. In *International Conference on Learning Representations*, 2026b.

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 *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.

Tianyu Zhao and Llion Jones. Fast-weight product key memory. 2026. URL https://arxiv.org/abs/2601.00671.

Table 4: Ablations of Triadic Gated DeltaNet at 400M parameters with a single seed. Top: first key dimension d_k versus second key dimension E at fixed joint key dimension $d_k \cdot E = 1024$, using a single forget gate per head. Bottom: activation function applied to the second key and query after the short convolution for $E = 8$.

Model	State	Params	0-shot ↑	Wiki. ↓	PG19 ≤4k ↓	PG19 4k–16k ↓	PG19 16k–64k ↓	Recall ↑
Triadic GDN (scalar gate), $d_k = 128$, $E = 8$	50.3	384.0	**53.7**	**10.80**	**14.77**	**14.01**	**13.70**	**33.4**
$d_k = 64$, $E = 16$	50.3	380.8	52.9	10.84	14.79	14.04	13.73	32.5
$d_k = 32$, $E = 32$	50.3	383.9	53.3	10.90	14.82	14.08	13.78	32.5
$d_k = 32$, $E = 32$, tied keys	50.3	380.7	52.1	10.89	14.81	14.09	13.79	31.9
Triadic GDN ($E = 8$), Softplus activation	50.3	385.4	53.5	10.86	**14.81**	**14.07**	**13.77**	32.2
Sigmoid activation	50.3	385.4	**53.7**	**10.86**	14.82	14.07	13.77	**33.3**
SiLU activation	50.3	385.4	53.0	10.98	14.92	14.16	13.84	32.8
Linear (no activation)	50.3	385.4	53.0	11.06	15.05	14.29	13.98	32.9

A Ablations

Key dimensions at fixed state size. In our main experiments, we keep the first key dimension d_k fixed at 128 and only vary the second key dimension from $E = 1$ to $E = 8$. Table 4 ablates this choice for Triadic GDN with $8\times$ the state. We vary the first key dimension d_k against the second key dimension E at a fixed joint key dimension $d_k \cdot E = 1024$, while keeping the value dimension and the number of heads fixed and using a single scalar forget gate per head. Since a smaller first key shrinks the total key and query projections, we match parameters by increasing the MLP width. As the two key dimensions become more similar, perplexity degrades slightly. Still, we believe that factorizations that require smaller total key and query projections could be relevant, in particular for mixture-of-experts architectures.

Second-key activation function. We further ablate the activation function of the second key and query after the short convolution. Table 4 compares four activation functions for Triadic GDN with $8\times$ the state. Non-negative activation functions (softplus, sigmoid) consistently outperform activation functions that can take negative values (SiLU, no activation) in perplexity. One explanation is that signed entries let a token read and write to different slices with opposite signs, so their contributions can cancel out when the second query aggregates the slices.

B Additional Related Work

Test-time training. Test-time training (TTT) interprets the in-context update of a recurrent state as learning under a self-supervised inner loss (Sun et al., 2025). For a linear inner model with squared reconstruction loss $\|\mathbf{S}_t k_t - v_t\|^2$, one gradient step per token recovers the delta rule (Schlag et al., 2021). DeltaProduct (Siems et al., 2025) instead performs multiple gradient descent steps per token, while MesaNet (von Oswald et al., 2025) solves the regression to optimality at every step. These methods remain close to conventional linear attention and can be compared at matched state size. In contrast, many TTT variants, including TTT-MLP (Sun et al., 2025), LaCT (Zhang et al., 2026b), deep neural memories (Behrouz et al., 2025; 2026), and TTT-E2E (Tandon et al., 2025) use significantly larger states than conventional linear attention blocks. Our results show that increasing state size substantially improves linear attention. Since a gradient step can itself be interpreted as a linear-attention update (Irie et al., 2022), some of the gains of TTT variants may therefore be a result of their larger state capacity, particularly at long context lengths.

C MQAR Setup

Each example of N key-value pairs yields a sequence with $2N$ tokens. The first N positions contain the N key-value pairs in random order. The next N positions contain the N queries (all keys corresponding to a key-value pair in the first N positions) and the model has to predict the corresponding value. To construct a sequence, keys and values are drawn without replacement from two distinct sets of 8k tokens. Each key-value pair is represented as a single token by concatenating the embedding of the key with the embedding of the value. A query is represented by concatenating the embedding of the key with zeros. Queries are read-only, meaning they never update the recurrent

Algorithm 1 Block with vanilla multi-head triadic linear attention (H heads, key/value dimension D, second-key dimension E)

▷ *Vanilla triadic linear attention*

$u \leftarrow \mathrm{RMSNorm}(x_t)$
for $h = 1, \ldots, H$ **do**
  $q, k, v \leftarrow W_q^h u,\ W_k^h u,\ W_v^h u \in \mathbb{R}^D$
  $q', k' \leftarrow W_{q'}^h u,\ W_{k'}^h u \in \mathbb{R}^E$
  $S_h \leftarrow S_h + k \otimes k' \otimes v$
  $o_h \leftarrow S_h \times_1 q \times_2 q'$
end for
$x_t \leftarrow x_t + W_o\,[\mathrm{RMSNorm}(o_1); \ldots; \mathrm{RMSNorm}(o_H)]$

▷ *SwiGLU MLP*

$u \leftarrow \mathrm{RMSNorm}(x_t)$
$x_t \leftarrow x_t + W_{\mathrm{down}}\big(\mathrm{SiLU}(W_{\mathrm{gate}} u) \odot W_{\mathrm{up}} u\big)$

state, which is the linear-attention equivalent of masking out other queries in the attention mask of a transformer. Every query thus reads the same accumulated state.

Our model architecture consists of a vanilla variant of triadic linear attention interleaved with SwiGLU MLPs (Shazeer, 2020). We provide pseudocode of the resulting block in Algorithm 1. The sequence mixer is a pure outer-product update of a third-order tensor, without any forgetting, convolutions, delta rule or non-linearities (beyond RMSNorm). We choose $d_{\text{model}} = 128$, $H = 4$ heads, key/value dimension $D = 16$, SwiGLU width 384, and a vocabulary size of 16k (8k key tokens, 8k value tokens). The second-key dimension E varies over $\{1, 2, 4, 8, 16\}$, which induces a per-head state tensor of shape $16 \times E \times 16$. $E = 1$ recovers ordinary (two-dimensional) linear attention. All models range from 3.51M to 3.54M parameters, where the majority (3.15M) sit in the embedding table and output head.

We train for 50k steps with a batch size of 250k tokens per step, yielding 12.5 billion tokens in total. Every step we sample $\lfloor 250\mathrm{k}/2N \rfloor$ fresh sequences of length $2N$. We use AdamW with learning rate 10^{-3}, $\beta = (0.9, 0.999)$, weight decay 0.1, cosine decay to 0, no gradient clipping, and bfloat16. We sweep $N \in \{32, 64, \ldots, 4096\}$ and $E \in \{1, 2, 4, 8, 16\}$ for 25 seeds per cell and present the mean test accuracy over all seeds, measured on a held-out set of 3k sequences, in Figure 1.

D Language Modeling Setup

Model architecture. All models consist of 24 pre-norm blocks with RMSNorm, a sequence mixer and a SwiGLU MLP. Algorithm 2 gives the GDN and sGLA blocks. Since the second query and key are l2-normalized, at $E = 1$ they are both 1 and thus the triadic variant reduces to the base mixer. The Transformer (Vaswani et al., 2017) uses grouped-query attention (Shazeer, 2019; Ainslie et al., 2023) with 8 query heads per key-value head, QK-norm (Dehghani et al., 2023) and RoPE (Su et al., 2021) with base theta 10k during pretraining and 2M for extension. We found that both QK-norm and the higher base frequency at extension (compared to 10k, 500k, 1M) strengthen the Transformer baseline. All models use the Llama-2 tokenizer (Touvron et al., 2023) with 32k vocabulary size.

Evaluation. We evaluate the final checkpoint of the long-context extension. PG19 perplexity is measured on around 1700 held-out books on windows of 64k tokens, which yields around 150M tokens in total. For WikiText, we report per-token perplexity on the WikiText-2 test set, with each document passed into the model in its entirety. The zero-shot score we report is the average accuracy over LAMBADA (Paperno et al., 2016), HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), ARC-Easy (Clark et al., 2018), ARC-Challenge (Clark et al., 2018), WinoGrande (Sakaguchi et al., 2020), OpenBookQA (Mihaylov et al., 2018), SciQ (Welbl et al., 2017), BoolQ (Clark et al., 2019) and COPA (Roemmele et al., 2011), obtained through the lm-eval harness (Gao et al., 2024). We use length-normalized accuracy for HellaSwag, ARC-Challenge and OpenBookQA, and plain accuracy for the other tasks. For needle-in-a-haystack retrieval, we use the eight NIAH tasks of RULER (Hsieh et al., 2024) at context lengths from 1k to 64k tokens and report the average over all tasks

Algorithm 2 Block with triadic GDN / sGLA as used in our language modeling experiments (H heads, key/value dimension D, second-key dimension E, state $S_h \in \mathbb{R}^{D\times E\times D}$ per head)

▷ *Triadic linear attention with forgetting and (for GDN) the delta rule*

$u \leftarrow \mathrm{RMSNorm}(x_t)$
for $h = 1, \dots, H$ **do**
  $q, k, v \leftarrow \mathrm{SiLU}(\mathrm{Conv}(W_q^h u)),\ \mathrm{SiLU}(\mathrm{Conv}(W_k^h u)),\ \mathrm{SiLU}(\mathrm{Conv}(W_v^h u)) \in \mathbb{R}^D$
  $q', k' \leftarrow \mathrm{softplus}(\mathrm{Conv}(W_{q'}^h u)),\ \mathrm{softplus}(\mathrm{Conv}(W_{k'}^h u)) \in \mathbb{R}^E$
  $q, k, q', k' \leftarrow q/\|q\|,\ k/\|k\|,\ q'/\|q'\|,\ k'/\|k'\|$
  $\alpha_e \leftarrow \exp\big(-\exp(A_e^h)\,\mathrm{softplus}(W_\alpha^{h,e} u + b_e^h)\big)$ for $e = 1, \dots, E$
  $\beta \leftarrow \sigma(W_\beta^h u)$
  $S_h \leftarrow S_h \times_2 \mathrm{diag}(\alpha)$
  if GDN **then**
    $S_h \leftarrow S_h + \beta\, k \otimes k' \otimes \big(v - S_h \times_1 k \times_2 k'\big)$
  else if sGLA **then**
    $S_h \leftarrow S_h + k \otimes \beta k' \otimes v$
  end if
  $o_h \leftarrow S_h \times_1 q \times_2 q'$
  $o_h \leftarrow \mathrm{RMSNorm}(o_h) \odot \sigma\big(W_{g,2}^h W_{g,1} u + b_g^h\big)$
end for
$x_t \leftarrow x_t + W_o\,[o_1; \dots; o_H]$

▷ *SwiGLU MLP*

$u \leftarrow \mathrm{RMSNorm}(x_t)$
$x_t \leftarrow x_t + W_{\mathrm{down}}\big(\mathrm{SiLU}(W_{\mathrm{gate}} u) \odot W_{\mathrm{up}} u\big)$

and lengths in Table 3. Figure 3 shows four representative tasks, where the model retrieves a single needle hidden in essays (niah_single_2), one of four needles with different keys (niah_multikey_1), all four of four needles (niah_multiquery), or four values stored under the same key (niah_multivalue). For recall, we use the six tasks of Arora et al. (2024), namely DROP, NQ, TriviaQA, FDA, SQuAD and SWDE, and score an answer as correct if the generation contains it.

E GPU KERNELS

Section 2.5 derives the chunkwise form of triadic linear attention without forget gates and the delta rule. Here we give the form with both, as Triadic GDN uses it, and describe the kernels that compute it.

Chunkwise form. Following Yang et al. (2023; 2024), we split the sequence into chunks of $C = 64$ positions indexed by i, write $\square^r = \square_{iC+r}$ for the r-th position of chunk i, and write $\mathbf{S}_{[i],e} \in \mathbb{R}^{d\times d}$ for slice e of the state at the start of chunk i. We drop the chunk index on all other quantities of the current chunk. Its keys, queries and values are stacked into $\mathbf{K}, \mathbf{Q}, \mathbf{V} \in \mathbb{R}^{C\times d}$, its second keys and queries into $\mathbf{K}', \mathbf{Q}' \in \mathbb{R}^{C\times E}$, whose columns $\mathbf{k}'_e, \mathbf{q}'_e \in \mathbb{R}^C$ belong to slice e, and its write strengths into $\boldsymbol{\beta} \in \mathbb{R}^C$. The decay that slice e accumulates over the first r positions of the chunk is $\gamma_e^r = \prod_{j=1}^r \alpha_e^j$, and $\boldsymbol{\gamma}_e \in \mathbb{R}^C$ collects it. As in Gated DeltaNet (Yang et al., 2025b), each slice has a decay matrix $\mathbf{\Gamma}_e \in \mathbb{R}^{C\times C}$ with $(\mathbf{\Gamma}_e)_{rs} = \gamma_e^r/\gamma_e^s$ for $r \geq s$ and 0 otherwise. The part of the joint key $\boldsymbol{\kappa}^r = \mathbf{k}^r \otimes \mathbf{k}'^r$ that addresses slice e is $k_e'^r \mathbf{k}^r$, and each slice decays by a single scalar, so the decayed inner product of two joint keys still separates into a term of the keys and a term of the second keys,

$$\sum_{e=1}^{E} \frac{\gamma_e^r}{\gamma_e^s} \big(k_e'^r \mathbf{k}^r\big)^\top \big(k_e'^s \mathbf{k}^s\big) = \big(\mathbf{k}^{r\top} \mathbf{k}^s\big)\, R_{rs}, \qquad \mathbf{R} = \sum_{e=1}^{E} \big(\mathbf{k}'_e \mathbf{k}'^\top_e\big) \odot \mathbf{\Gamma}_e, \qquad r \geq s. \tag{9}$$

The UT transform of the chunkwise delta rule is therefore formed from products of the d-dimensional keys and the $C^2 E$ terms of $\mathbf{R}$, rather than from the $d \cdot E$-dimensional joint keys,

$$\mathbf{T} = \Big(\mathbf{I} + \mathrm{tril}\big(\mathrm{diag}(\boldsymbol{\beta})\,(\mathbf{K}\mathbf{K}^\top \odot \mathbf{R}),\, -1\big)\Big)^{-1} \mathrm{diag}(\boldsymbol{\beta}), \tag{10}$$

and it is the same $C \times C$ matrix for all E slices. With $\mathbf{R}' = \sum_e (\mathbf{q}'_e \mathbf{k}'^{\top}_e) \odot \mathbf{\Gamma}_e$ for the query side, one chunk computes

$$\begin{aligned} \mathbf{U} &= \mathbf{T}\Big(\mathbf{V} - \textstyle\sum_e \operatorname{diag}(\mathbf{k}'_e \odot \boldsymbol{\gamma}_e)\, \mathbf{K}\mathbf{S}_{[i],e}\Big), \\ \mathbf{O} &= \textstyle\sum_e \operatorname{diag}(\mathbf{q}'_e \odot \boldsymbol{\gamma}_e)\, \mathbf{Q}\mathbf{S}_{[i],e} + \big(\mathbf{Q}\mathbf{K}^{\top} \odot \mathbf{R}'\big)\, \mathbf{U}, \\ \mathbf{S}_{[i+1],e} &= \gamma_e^C\, \mathbf{S}_{[i],e} + \mathbf{K}^{\top} \operatorname{diag}\big(\mathbf{k}'_e \odot \gamma_e^C / \boldsymbol{\gamma}_e\big)\, \mathbf{U}, \end{aligned} \tag{11}$$

where the division acts elementwise and $\mathbf{U}$ holds the chunk's delta-corrected values. Without forget gates, every $\mathbf{\Gamma}_e$ is the causal mask, and $\mathbf{R}'$ reduces to $\operatorname{tril}(\mathbf{Q}'\mathbf{K}'^{\top})$ of Section 2.5; for $E = 1$ and $\mathbf{k}' = \mathbf{q}' = \mathbf{1}$, $\mathbf{R} = \mathbf{R}' = \mathbf{\Gamma}$ and Equation 11 is the chunkwise form of Gated DeltaNet. Unlike DeltaNet, which precomputes $\mathbf{W} = \mathbf{T}\mathbf{K}$ and $\mathbf{T}\mathbf{V}$ before the recurrence (Yang et al., 2024), we apply $\mathbf{T}$ to the residual, since with per-slice decay the analogue of $\mathbf{W}$ would be one matrix $\mathbf{T} \operatorname{diag}(\mathbf{k}'_e \odot \boldsymbol{\gamma}_e)\mathbf{K}$ per slice.

Numerical stability. The per-slice gates make the decay vary along the joint key, which in gated linear attention requires a secondary level of chunking (Yang et al., 2023). Here each slice decays by a single scalar, so the decay enters only the C^2E terms of $\mathbf{R}$ and $\mathbf{R}'$ and the per-slice scalings in Equation 11, and the kernels evaluate each decay factor, such as γ_e^r/γ_e^s, directly as the exponential of a difference of accumulated log gates. Every such difference is non-positive, because the log decay only decreases within a chunk and $\mathbf{\Gamma}_e$ is used only for $r \geq s$. The computation therefore cannot overflow at any decay rate, and the gates need no clamp. This rules out a cheaper construction of $\mathbf{R}$. Writing each factor as $(\gamma_e^r/m_e)(m_e/\gamma_e^s)$ around a reference value m_e in the middle of the chunk turns $\mathbf{R}$ into the lower triangle of a product of two $C \times E$ matrices and reduces the number of exponentials from C^2E to $2CE$, but gives the factors positive exponents, which overflow in FP32 above about 88.7. With the exponents clamped to $[-88, 88]$, once a slice decays by more than a factor of e^{176} within one chunk, the clamp distorts the near-diagonal entries of $\mathbf{R}$. In an FP64 simulation of one chunk that emulates only this clamp, the relative error of the chunk's output is 0.05 at a within-chunk decay of e^{-200} and 1.5 at e^{-600}. We therefore form $\mathbf{R}$ and $\mathbf{R}'$ element by element.

Input projections. The second keys and queries share the projection and the short convolution of the queries, keys and values: one GEMM produces $\mathbf{q}, \mathbf{k}, \mathbf{v}, \mathbf{k}'$ and $\mathbf{q}'$, and one causal depthwise convolution processes all of them, with SiLU on the channels of $\mathbf{q}$, $\mathbf{k}$ and $\mathbf{v}$ only. A triadic layer therefore runs the same two projection GEMMs and one convolution as a GDN layer, with wider projections.

Kernels. We implement the chunkwise form for Hopper GPUs in the CuTe DSL of CUTLASS (Thakkar et al., 2023). The forward pass accumulates the log decays, forms $\mathbf{R}$ and $\mathbf{R}'$, computes the inverse in $\mathbf{T}$ with a blockwise triangular solve accumulated in FP32, and then runs one recurrent kernel over the chunks of each sequence, with one thread block per document, head and value block of 32 columns, or 16 when a step holds too few documents to fill the GPU. This kernel is warp-specialized. At $E = 8$, two warpgroups hold four state slices each and compute their products with asynchronous tensor-core instructions, a third warpgroup loads each chunk's inputs through the Tensor Memory Accelerator into double-buffered shared memory and forms $\mathbf{U}$, and a fourth forms the output. For training, the forward pass also stores the state at the start of every chunk in BF16, so that the backward pass does not recompute it. The backward pass runs a short reverse recurrence over the chunks that stores the gradient of each state slice, after which all remaining gradients are computed for every chunk in parallel. Every reduction follows a fixed order without floating-point atomics, so the gradients are bitwise reproducible. Documents packed into one training sequence each run their own recurrence from a zero state, directly on the packed tensors. We verify the output and the gradients of all seven inputs against FP64 references at the 1.3B training shape, with a relative ℓ_2 error below 1%.

F Analysis

Long-context utilization. A larger recurrent state is useful when it enables the model to preserve information from long ago. We test this directly on our trained long-context checkpoints of GDN

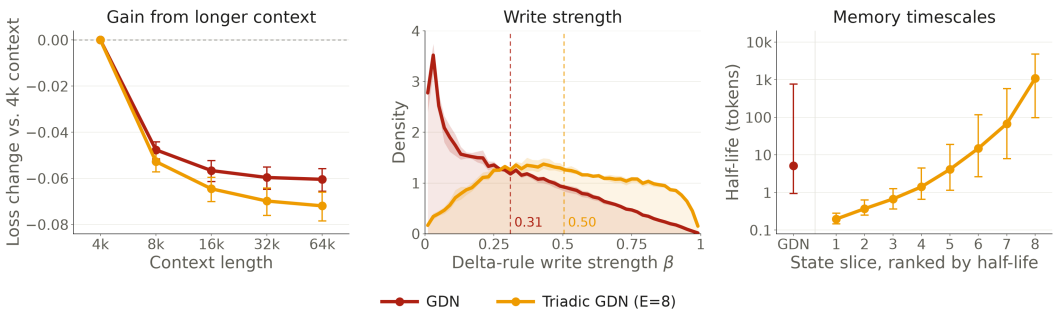

Figure 5: How Triadic GDN (E=8) uses the larger state compared to GDN. Left: change in loss on the final 4k tokens of a book when increasing the preceding context from 4k to 64k. Middle: distribution of delta strengths β. Dashed lines mark the means. Right: half-life of each state slice, ranked within its head.

and Triadic GDN (E=8) across three seeds on held-out PG19 books containing at least 64k tokens. Per book, we fix the prediction target to the last 4k tokens and vary the context the state reads before predicting them. As shown in Figure 5 (left), Triadic GDN gains consistently more from added context than vanilla Gated DeltaNet. In addition, we find that GDN begins to saturate by 16k context tokens, while Triadic GDN continues improving through 32k.

Memory update dynamics. We next examine how enlarging the state size changes how the model works with its memory. When comparing Triadic GDN (E=8) against GDN, we find that Triadic GDN learns substantially larger delta-rule write strengths β than GDN, with the per-token β distribution also becoming more broadly distributed. Averaged across all evaluated tokens, heads, and layers, the mean increases from 0.31 to 0.50 (Figure 5, middle). We also find that the additional state slices develop a broad range of memory timescales. To quantify this, we measure each slice's half-life, or the number of tokens over which the forget gates reduce an existing state contribution by half. Ranking the eight slices within each head by this half-life (Figure 5, right), the median half-life ranges from 0.20 tokens for the shortest-lived slice to 1080 tokens for the longest-lived slice. In comparison, GDN only has a single forget gate per head, with a median half-life of 5.1 tokens. Together, these suggest that the additional state axis permits the model to make stronger memory updates while keeping information over a wider range of timescales.