RUNNINGTENSOR: GENERALIZING LINEAR ATTENTION TO HIGHER-ORDER RECURRENT STATES

Anonymous authors
Paper under double-blind review

ABSTRACT

Linear attention and state-space models provide linear-time sequence modeling, but their recurrent memory remains a second-order tensor, a matrix, limiting the order of interactions that can be represented in the state. We introduce the RunningTensor, which generalizes this memory to an order-o tensor, updated by a rank-1 outer product and read by contracting against $o-1$ vector queries. Order 2 recovers linear attention; we study order 3 as a proof of concept, retaining both recurrent and parallel forms while remaining linear in sequence length T and improving working memory capacity from $\mathcal{O}(W^2)$ to $\mathcal{O}(W^o)$. On synthetic multi-query associative recall, RunningTensor outperforms linear attention and SSM baselines. On real multivariate classification, long-horizon forecasting, and 16 kHz speech, a reduced RT/4 tensor remains competitive with linear baselines at long sequence length.

1 INTRODUCTION

Self-attention in Transformers Radford et al. (2018; 2019); Vaswani et al. (2017) retrieves from the past by materializing pairwise scores between all token positions. The cost is well-documented: that score matrix grows as $\mathcal{O}(T^2)$ in time and memory, which makes long-sequence processing computationally prohibitive. Linearized attention Katharopoulos et al. (2020) and *state-space models* (SSMs) such as S4 Gu et al. (2022), Mamba Gu & Dao (2024), Gated DeltaNet Yang et al. (2025b), and Longhorn Liu et al. (2025) recast the same retrieval as a linear recurrence with input-dependent gating. The formulation is dual: at inference the model ticks forward one step at a time in $\mathcal{O}(1)$ with respect to sequence length T; at training the recurrence unrolls into a matrix multiplication and can be parallelized. This removes the quadratic bottleneck in T, but it does not change the geometry of the memory. Linear attention accumulates information into a matrix state and reads it through a single vector contraction. The summary of the past is therefore bounded by the size and rank of that matrix, which limits memory and expressivity of linearized attention. The question we ask is whether the recurrent state can be lifted beyond this second-order accumulation and single query, without giving up linear cost in T.

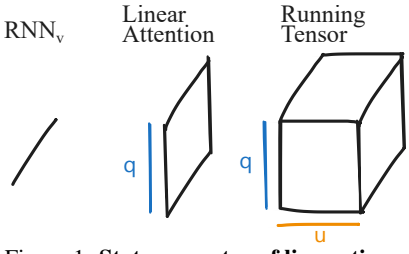

Figure 1: **State geometry of linear-time recurrences.** A vector-state RNN (RNN$_v$) has linear memory capacity, linear attention has quadratic memory capacity given its matrix state, and RunningTensor can increase state size and therefore memory capacity to cubic and higher. One vector query q contraction turns a matrix state into a vector state, but several vector queries q, u are required to turn a tensor state into a vector output.

We propose the *RunningTensor*, a causal sequence model whose state is a tensor of order $o \geq 2$, that is updated by a rank-1 outer product and read by contracting against several vector queries. The reason to raise the order, rather than widen a matrix until it has the same number of slots, is to buy associative capacity at fixed width: more addressable bindings, and a two-key readout, without growing model width. This work instantiates $o = 3$ as a proof of concept, and the construction maps to linear attention when $o = 2$. Proposition 1 shows that linear attention is contained as a special case, so RunningTensor is strictly more expressive once the extra modes are used. Writing W for the width of each state mode, state size then follows a hierarchy indexed by tensor order: classic vector-state RNNs Cho et al. (2014); Hochreiter & Schmidhuber (1997) use $\mathcal{O}(W)$ memory, matrix-state linear attention uses $\mathcal{O}(W^2)$, and an order-o RunningTensor uses $\mathcal{O}(W^o)$, while a full

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.

pass remains linear in T. The instantiated model adds data-dependent forget gates to reweight past outer products, query-skip gates that let a readout use both tensor axes, one, or none, and ConcOnlyC, which replaces the fused linear projection by a near-parameter-free channel mix.

Our contributions are as follows:

- We propose the *RunningTensor*, a linear-time causal sequence model whose state is a tensor of order $o \geq 2$. Order 2 recovers linear attention; this work instantiates order 3.
- On a synthetic multi-query recall benchmark, RunningTensor improves over linear attention and SSM baselines.
- On real multivariate classification, long-horizon forecasting, and 16 kHz speech, a reduced RT/4 tensor remains competitive with linear baselines at long sequence length.

2 Methodology: The RunningTensor

2.1 Preliminary: Linear Attention

The linear transformer Katharopoulos et al. (2020) can be written as a linear recurrence when normalization and query/key feature maps are omitted:

$$\mathbf{S}_t = \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top \in \mathbb{R}^{d_v \times d_k}, \qquad \mathbf{o}_t = \mathbf{S}_t \mathbf{q}_t \in \mathbb{R}^{d_v},$$

where d_k and d_v denote the per-head dimensions of the query/key and value vectors, $\mathbf{q}_t, \mathbf{k}_t^\top, \mathbf{v}_t$ are usually linear projections of an input representation $\mathbf{x}_t$, $\mathbf{S}_t$ is a matrix-shaped recurrent state and $\mathbf{o}_t$ the output of the unit. Unrolling the recurrence yields equivalent vector and matrix forms. The sum is causal, which at the element level is a lower-triangular mask $M_{ti} = \mathbf{1}_{i \leq t}$:

$$\mathbf{o}_t = \sum_{i=1}^{t} (\mathbf{v}_i \mathbf{k}_i^\top) \mathbf{q}_t = \sum_{i=1}^{t} \mathbf{v}_i \big(\mathbf{k}_i^\top \mathbf{q}_t\big) = \sum_{i=1}^{T} M_{ti}\, \mathbf{v}_i \big(\mathbf{k}_i^\top \mathbf{q}_t\big) \in \mathbb{R}^{d_v},$$
$$\implies \mathbf{O} = \big(\mathbf{Q}\mathbf{K}^\top \odot \mathbf{M}\big)\, \mathbf{V} \in \mathbb{R}^{T \times d_v},$$

where $\mathbf{Q}, \mathbf{K} \in \mathbb{R}^{T \times d_k}$, $\mathbf{V} \in \mathbb{R}^{T \times d_v}$, $\mathbf{M} \in \mathbb{R}^{T \times T}$ is the causal mask and $\odot$ denotes element-wise multiplication. This formulation highlights the dual perspective: a recurrent structure enabling linear-time inference, and a matrix form that allows efficient parallel training.

2.2 The RunningTensor architecture

The RunningTensor. To find a tensor analogue, we define a third-order state using the tensor outer product $\otimes$:

$$\boldsymbol{\mathcal{S}}_t = \boldsymbol{\mathcal{S}}_{t-1} + \mathbf{v}_t \otimes \mathbf{k}_t \otimes \mathbf{r}_t \in \mathbb{R}^{d_v \times d_k \times d_r}$$

with $\mathbf{v}_t \in \mathbb{R}^{d_v}, \mathbf{k}_t \in \mathbb{R}^{d_k}, \mathbf{r}_t \in \mathbb{R}^{d_r}$. The same construction extends to order $o > 3$: accumulate a rank-1 outer product of o vectors and contract against $o - 1$ queries. We restrict to $o = 3$ for simplicity.

Contracting modes 2 and 3 with queries $\mathbf{q}_t$ and $\mathbf{u}_t$ gives:

$$(o_t)_a = \sum_{b=1}^{d_k} \sum_{c=1}^{d_r} (\mathcal{S}_t)_{abc}\, (q_t)_b\, (u_t)_c, \qquad a = 1, \ldots, d_v.$$

with $\mathbf{q}_t \in \mathbb{R}^{d_k}$, $\mathbf{u}_t \in \mathbb{R}^{d_r}$. Expanding the recurrence:

$$\mathbf{o}_t = \sum_{i=1}^{t} \mathbf{v}_i\, (\mathbf{k}_i^\top \mathbf{q}_t)\, (\mathbf{r}_i^\top \mathbf{u}_t) = \sum_{i=1}^{T} M_{ti}\, \mathbf{v}_i\, (\mathbf{k}_i^\top \mathbf{q}_t)\, (\mathbf{r}_i^\top \mathbf{u}_t) \in \mathbb{R}^{d_v}.$$
$$\implies \mathbf{O} = \big((\mathbf{Q}\mathbf{K}^\top) \odot (\mathbf{U}\mathbf{R}^\top) \odot \mathbf{M}\big)\, \mathbf{V} \in \mathbb{R}^{T \times d_v}.$$

Proposition 1 (Linear attention as a special case)**.** *If $d_r = 1$ and $\mathbf{u}_t = \mathbf{r}_t = 1$ for all t, then the RunningTensor matrix form reduces to causal linear attention.*

Proof. With $d_r = 1$ and $\mathbf{u}_t = \mathbf{r}_t = 1$, one has $\mathbf{r}_i^\top \mathbf{u}_t = 1$ and $(\mathbf{U}\mathbf{R}^\top)_{ti} = 1$. The expanded readout becomes $\mathbf{o}_t = \sum_{i=1}^{T} M_{ti}\, \mathbf{v}_i(\mathbf{k}_i^\top \mathbf{q}_t)$, and the matrix form collapses to $\mathbf{O} = (\mathbf{Q}\mathbf{K}^\top \odot \mathbf{M})\,\mathbf{V}$. □

The RunningTensor is therefore strictly more expressive than linear attention. Proposition 1 embeds causal linear attention as a special case ($d_r = 1$, trivial second query). Whenever $d_r > 1$ and both query axes are active, the readout multiplies two key-query scores at each past step, so the recurrence implements bilinear addressing that no order-2 matrix state can match without collapsing its keys into a single d_k-dimensional slot.

Forget gates (α). Without decay, every past outer product enters the state with equal weight. As in several linear attention alternatives, we add data-dependent forgetting so earlier context can be weighted. Per head, meaning several times in parallel, we have

$$\boldsymbol{\mathcal{S}}_t = \alpha_t \boldsymbol{\mathcal{S}}_{t-1} + \mathbf{v}_t \otimes \mathbf{k}_t \otimes \mathbf{r}_t.$$

Unrolling introduces $\gamma_j = \prod_{i=1}^{j} \alpha_i$ and $\Gamma_{ts} = \gamma_t/\gamma_s$. In log space, $\Gamma_{ts} = \exp(g_t - g_s)$ with $g_t = \log \gamma_t$, which is the factor used in the bilinear form. We parameterize g following Gated Linear Attention (GLA) Yang et al. (2024b): from a raw per-head logit a_t produced on the fused input path,

$$g_t = \frac{1}{\tau} \log\sigma(a_t) = \frac{1}{\tau} \log \frac{1}{1 + e^{-a_t}}, \qquad \tau = 16,$$

so $g_t \in (-\infty, 0]$. The per-step multiplier is then $\alpha_t = \exp(g_t - g_{t-1}) = \big(\sigma(a_t)/\sigma(a_{t-1})\big)^{1/\tau}$, which may exceed 1, in contrast to forget gates constrained to $[0, 1]$. Expanding the sum gives the vector form and, stacking time rows, the parallel matrix form (per head):

$$\mathbf{o}_t = \sum_{i=1}^{t} \mathbf{v}_i \Big(\Gamma_{ti}\, \mathbf{k}_i^\top \mathbf{q}_t\, \mathbf{r}_i^\top \mathbf{u}_t \Big) = \sum_{i=1}^{T} M_{ti}\, \Gamma_{ti}\, \mathbf{v}_i \big(\mathbf{k}_i^\top \mathbf{q}_t\big)\big(\mathbf{r}_i^\top \mathbf{u}_t\big),$$
$$\Longrightarrow \quad \mathbf{O} = \Big((\mathbf{Q}\mathbf{K}^\top) \odot (\mathbf{U}\mathbf{R}^\top) \odot \mathbf{M} \odot \boldsymbol{\Gamma} \Big)\, \mathbf{V}.$$

Query-skip gates (ξ_i). The bilinear score $(\mathbf{k}_s^\top \mathbf{q}_t)(\mathbf{r}_s^\top \mathbf{u}_t)$ always queries both axis of the state. We provide the network the ability to decide when to query both axis, one, or none, resulting in a score of the form

$$\Big((1 - \xi_{1,t}) + \xi_{1,t}\, \mathbf{k}_s^\top \mathbf{q}_t \Big) \Big((1 - \xi_{2,t}) + \xi_{2,t}\, \mathbf{r}_s^\top \mathbf{u}_t \Big), \qquad \xi_{1,t}, \xi_{2,t} \in (0, 1),$$

where $\xi_{i,t}$ are per-head *query-skip gates* at time t, and $i \in \{1, 2\}$. Rather than evaluating the mixed score directly, we embed it in the existing recurrence via a homogeneous feature map: dummy slots are appended so that

$$\hat{\mathbf{q}}_t = [(1 - \xi_{1,t}),\ \xi_{1,t}\mathbf{q}_t], \qquad \hat{\mathbf{k}}_s = [1,\ \mathbf{k}_s], \qquad \hat{\mathbf{u}}_t = [(1 - \xi_{2,t}),\ \xi_{2,t}\mathbf{u}_t], \qquad \hat{\mathbf{r}}_s = [1,\ \mathbf{r}_s],$$

and the delta or additive core runs on $(\hat{\mathbf{q}}, \hat{\mathbf{k}}, \hat{\mathbf{u}}, \hat{\mathbf{r}})$. In the full model we additionally ℓ_2-normalize the expanded queries and keys so pairwise dot products remain bounded, improving delta rule stability. When $\xi_{1,t} = 1$ and $\xi_{2,t} = 0$, the score collapses to $\mathbf{k}_s^\top \mathbf{q}_t$ and the readout coincides with causal linear attention on the $(\mathbf{k}, \mathbf{q})$ mode; other gate settings activate the second axis or a skip constant. Therefore, the RunningTensor strictly contains the linear attention function class also in this second way, and is therefore strictly more expressive.

Delta rule Δ (β). For the full model we replace pure accumulation with the *tensor delta rule*, analogous to DeltaNet Irie et al. (2021); Yang et al. (2024c) and Gated DeltaNet Yang et al. (2025b), lifted to third order. Let $\langle \boldsymbol{\mathcal{S}}, \mathbf{k}, \mathbf{r} \rangle$ denote mode-wise contraction of $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{d_v \times d_k \times d_r}$ with $\mathbf{k}$ and $\mathbf{r}$. With forget multiplier α_t as above, the recurrent update is

$$\boldsymbol{\mathcal{S}}_t = \alpha_t \boldsymbol{\mathcal{S}}_{t-1} + \beta_t \Big(\mathbf{v}_t - \big\langle \alpha_t \boldsymbol{\mathcal{S}}_{t-1}, \hat{\mathbf{k}}_t, \hat{\mathbf{r}}_t \big\rangle \Big) \otimes \hat{\mathbf{k}}_t \otimes \hat{\mathbf{r}}_t,$$
$$\mathbf{o}_t = \big\langle \boldsymbol{\mathcal{S}}_t, \hat{\mathbf{q}}_t, \hat{\mathbf{u}}_t \big\rangle.$$

The subtracted prediction $\langle \alpha_t \boldsymbol{\mathcal{S}}_{t-1}, \hat{\mathbf{k}}_t, \hat{\mathbf{r}}_t \rangle$ removes the component of $\mathbf{v}_t$ already represented along $(\hat{\mathbf{k}}_t, \hat{\mathbf{r}}_t)$, so the scalar overwrite gate β_t controls step size. Training uses a chunk-parallel UT/WY implementation of this recurrence; inference unrolls the same rule step by step.

ConcOnlyC de-parameterization. By default, separate linear projections would map the layer input $\mathbf{x}_t \in \mathbb{R}^{d_{\text{model}}}$ to its query, key, value, and gates. It is instead common to produce all of these features in parallel with a single linear layer. The fused output is a wide vector of length d_{fused} that is then split into the individual features. Let $\mathbf{x} \in \mathbb{R}^{T \times d_{\text{model}}}$; the fused width is

$$d_{\text{fused}} = n_h\big(2d_k + 2d_r + 2d_v + 4\big),$$

for $(\mathbf{q}, \mathbf{k}, \mathbf{u}, \mathbf{r}, \mathbf{v}, a, b, z_1, z_2, \mathbf{g}^{\text{out}})$ across n_h heads, where a is the temporal gate, b the delta overwrite gate, z_1, z_2 the raw query-skip channels, and $\mathbf{g}^{\text{out}} \in \mathbb{R}^{n_h d_v}$ the output gate. Gated DeltaNet Yang et al. (2025b) implements this fusion with one dense matrix in $\mathbb{R}^{d_{\text{model}} \times d_{\text{fused}}}$. We replace that matrix with what we name *ConcOnlyC*: concatenate $\mathbf{x}$ with itself along the width axis, to reach the desired width, and use a 3 parameters convolution moving along the width axis after, to mix the channels. Mathematically,

$$\tilde{\mathbf{x}}_{t,:} = \big[\mathbf{x}_{t,:},\ \mathbf{x}_{t,:},\ \ldots\big]_{1:d_{\text{fused}}}$$

$$\text{ConcOnlyC}(\mathbf{x})_{t,c} = \sum_{k=-(K_{width}-1)/2}^{(K_{width}-1)/2} w_k\, \tilde{\mathbf{x}}_{t,\,c+k}.$$

with kernel $K_{width} = 3$ and weights $\mathbf{W}_{width} \in \mathbb{R}^{K_{width}}$. This replaces the fused linear, and $d_{\text{model}} \cdot d_{\text{fused}}$ parameters become 3. We then implement a per-channel causal convolution of kernel $K_{time} = 4$ and weights $\mathbf{W}_{time} \in \mathbb{R}^{K_{time} \times d_{\text{fused}}}$ along time on the $(\mathbf{q}, \mathbf{k}, \mathbf{u}, \mathbf{r}, \mathbf{v}, a, b, z_1, z_2, \mathbf{g}^{\text{out}})$ block. Gated DeltaNet instead, only applies the temporal convolution to $(\mathbf{q}, \mathbf{k}, \mathbf{v})$. After the split, $\mathbf{q}, \mathbf{k}, \mathbf{u}, \mathbf{r}, \mathbf{v}$ pass through SiLU, and $\mathbf{q}, \mathbf{k}, \mathbf{u}, \mathbf{r}$ are ℓ_2-normalized per head before the query-skip feature map expands and re-normalizes them; $\mathbf{v}$ is not normalized. The temporal logits a become g via $\log\sigma/\tau$ as above; the overwrite gate is $\beta = \sigma(b)$ and the query-skip gates are $\xi_i = \sigma(z_i)$. We do not apply $1/\sqrt{d}$ attention scaling since the ℓ_2-normalization is already playing that role.

The RunningTensor Block. As in Gated DeltaNet Yang et al. (2025b), we apply a multiplicative output gate after the recurrence via FusedRMSNormGated. In their configuration, the gate logits bypass the short convolution so local mixing does not blur them across time. In the full RunningTensor instead, the gate undergoes the same ConcOnlyC channel mix and causal short convolution as the queries and the other gates, yielding a time-local, context-aware modulation of the readout. After the delta (or additive bilinear) core returns $\mathbf{o} \in \mathbb{R}^{T \times n_h \times d_v}$, FusedRMSNormGated applies $\text{RMSNorm}(\mathbf{o}) \odot \text{SiLU}(\mathbf{g}^{\text{out}})$, then a linear map $n_h d_v \to d_{\text{model}}$ produces the layer output. The decoder residual around the layer is added outside this map. The RunningTensor sits in a Pre-LN decoder block of the same form as Llama Touvron et al. (2023), Qwen2/3 Yang et al. (2024a; 2025a), and Mistral Jiang et al. (2023), replacing self-attention.

3 Results

We first isolate *why* RunningTensor uses two queries on a matched-memory conjunctive recall example. We then test associative recall at scale on MQAR at long context, ablate architectural choices (input fusion, query-skip gates, delta rule, output gating, and tensor vs. matrix state), and report the same stack on real time-series classification, forecasting, and raw-audio recognition.

3.1 Why two queries?

What changes if we keep the memory capacity of a RunningTensor equal to the one of a linear attention, what do we gain with that extra query? In Figure 2 we compare how both would tackle the retrieval of four memories that are built tying together two conjunctive keys: $(\text{red}, \square) \mapsto \mathbf{v}_1$, $(\text{red}, \circ) \mapsto \mathbf{v}_2$, $(\text{blue}, \square) \mapsto \mathbf{v}_3$, $(\text{blue}, \circ) \mapsto \mathbf{v}_4$. A read action looks for *two* attributes, e.g. "blue" and "square"; the correct output is $\mathbf{v}_3$. We compare a matrix state and a third-order state with the *same* value width and the *same* number of scalar slots per head:

	RunningTensor	Linear attention
State shape	$d_v \times d_k \times d_r$	$d_v \times d_k$
Memory slots	$2 \cdot 2 \cdot 2 = 8$	$2 \cdot 4 = 8$
Key/query/value width	$\mathbf{k}, \mathbf{r}, \mathbf{q}, \mathbf{u} \in \mathbb{R}^2, \mathbf{v} \in \mathbb{R}^2$	$\mathbf{k}, \mathbf{q} \in \mathbb{R}^4, \mathbf{v} \in \mathbb{R}^2$

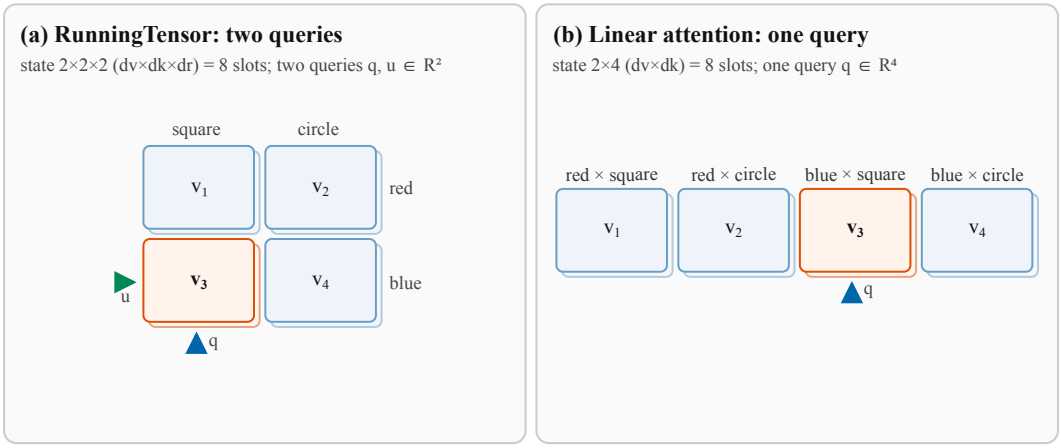

Figure 2: **Same eight slots; conjunctive lookup factorized vs. fused.** Both panels implement the same 2×2 task (four pair bindings, $d_v = 2$) at eight scalar state slots per head. **(a)** RunningTensor uses a $2 \times 2 \times 2$ state; the grid is the 2×2 address on $(\mathbf{k}, \mathbf{r})$, with $\mathbf{q}, \mathbf{u} \in \mathbb{R}^2$ selecting column and row. The product $(\mathbf{k}^\top \mathbf{q})(\mathbf{r}^\top \mathbf{u})$ retrieves $\mathbf{v}_3$ for blue $\times$ square. **(b)** Linear attention spends the same budget as a 2×4 matrix: two d_v rows and four columns, each column a combined key $\mathbf{k}_i \in \mathbb{R}^4$ for one pair. The four labels are the conjunctive targets; a single $\mathbf{q} \in \mathbb{R}^4$ must select among them at read time.

RunningTensor (two queries). Each write is $\mathbf{v} \otimes \mathbf{k} \otimes \mathbf{r}$ with $\mathbf{v} \in \mathbb{R}^2$, color factor $\mathbf{k} \in \mathbb{R}^2$, shape factor $\mathbf{r} \in \mathbb{R}^2$. Read uses $\mathbf{q}$ (color) and $\mathbf{u}$ (shape) with weight $(\mathbf{k}_i^\top \mathbf{q})(\mathbf{r}_i^\top \mathbf{u})$, column $\times$ row in the figure, then contract onto $\mathbf{v}_i \in \mathbb{R}^2$. For "blue" $\times$ "square" only the matching write gets a large product.

Linear attention (one query). The same four bindings use rank-one updates $\mathbf{v}\mathbf{k}^\top$ with $\mathbf{v} \in \mathbb{R}^2$ and one combined key $\mathbf{k} \in \mathbb{R}^4$ per pair. We therefore have four conjunctive labels, but a single $\mathbf{q} \in \mathbb{R}^4$ at read time and scores $\mathbf{k}_i^\top \mathbf{q}$. With $d_k = 4$, four keys can in principle separate the four pair-queries, unlike the tighter $d_k = 2$.

Here the comparison is structural at matched d_v and matched eight slots: the matrix must *mix* color and shape into each $\mathbf{k}_i$, while RunningTensor keeps them on separate axes and multiplies two 2-dimensional matches. That factorization is what we expect to help when queries are compositional. The third-order state does not add memory slots in this match; it factorizes the address into two queryable axes, which favors compositional bindings, and can yield more disentangled key factors when the task rewards conjunctive recall. We next leave this matched-slot regime and ask whether RunningTensor third-order state with its full slot budget provides an advantage.

3.2 Synthetic retrieval (MQAR)

We evaluate on multi-query associative recall (MQAR) Arora et al. (2024) at $d_{\text{model}} = 128$ with validation sequence lengths 1024, 2048, and 4096 tokens. Prior MQAR studies often stay in an easier band, with hidden width up to 512 and sequence length up to 512; we keep width small and push length, to test whether the extra tensor modes still help when recall is harder. Each cell reports the best validation F1 in % over 3 seeds and learning rates $\{10^{-2}, 10^{-3}, 10^{-4}\}$. Table 1 compares attention and SSM baselines to RunningTensor (ours).

At matched d_{model} and fewer parameters than attention, RunningTensor matches perfect MQAR F1 across all evaluated lengths while remaining competitive in forward-pass latency, but faster than attention and faster than the best SSM. Matrix-state and second-order baselines (Gated Δ Net, Mamba-2, GLA) degrade sharply as context grows; Δ Net retains strong performance but falls short of the tensor delta variant at 2048 tokens.

Table 1: MQAR validation F1 in % at hidden width 128. Attention, SSM baselines, and RunningTensor. Each cell reports the best validation F1 over 3 seeds and 3 learning rates. Columns 1024, 2048, and 4096 are MQAR validation sequence lengths in tokens. ms/step is mean validation forward-pass milliseconds per eval batch step; lower is faster. Params is non-embedding param count per layer. Avg is the mean over 1024, 2048, and 4096.

Method	**Params**	**ms/step**	**1024**	**2048**	**4096**	**Avg ↑**
Attention	934k	333.5	**100.0**	**99.9**	**100.0**	**100.0**
Gated Δ Net	249k	261.0	78.3	57.2	48.1	61.2
Mamba-2	284k	252.5	0.6	0.1	0.0	0.2
GLA	216k	267.2	0.1	1.9	0.0	0.7
Δ Net	249k	355.7	98.5	94.5	**100.0**	97.7
RunningTensor (ours)	176k	311.9	**100.0**	**100.0**	**100.0**	**100.0**

3.3 Component ablation

Table 2 reports RunningTensor variants at $d_{\text{model}} = 128$ under the same protocol as Table 1. GDN does not pass $\mathbf{g}^{\text{out}}$ through the temporal filter; when we do, as in the full architecture, we mark it $\mathbf{g}^{\text{out}}$@t.

Table 2: RunningTensor variants on MQAR validation F1 in % at $d_{\text{model}} = 128$. Same protocol as Table 1. Row labels match internal experiment summaries.

Method	**Params**	**ms/step**	**1024**	**2048**	**4096**	**Avg ↑**
RT w. Tensor S_t + ConcOnlyC + ξ_i + Δ + $\mathbf{g}^{\text{out}}$@t	176k	311.9	**100.0**	**100.0**	**100.0**	**100.0**
RT w. Tensor S_t + ConcOnlyC + Δ + $\mathbf{g}^{\text{out}}$@t	176k	322.0	99.9	**100.0**	**100.0**	**100.0**
RT w. Tensor S_t + ConcOnlyC + ξ_i + $\mathbf{g}^{\text{out}}$@t	176k	257.2	98.9	99.0	97.9	98.6
RT w. Tensor S_t + ConcOnlyC + Δ	175k	313.7	99.7	**100.0**	**100.0**	99.9
RT w. Tensor S_t + ConcOnlyC	175k	284.7	99.4	96.2	99.8	98.5
RT w. Tensor S_t + ConcOnly	175k	275.4	81.5	75.0	81.3	79.3
RT w. Tensor S_t + OneLinear	275k	254.2	97.2	98.2	54.2	83.2
RT w. Matrix S_t + ConcOnlyC	175k	281.0	41.4	22.7	27.7	30.6
RT w. Matrix S_t + ConcOnly	175k	268.6	44.3	4.9	13.0	20.7
RT w. Matrix S_t + OneLinear	249k	260.6	51.4	10.0	19.0	26.8

The first two blocks use the tensor recipe. The full model, ConcOnlyC with query-skip gates ξ_i, the delta rule, and $\mathbf{g}^{\text{out}}$@t, is perfect at every length. Dropping ξ_i still averages 100. Dropping the delta rule costs a little at 4096 (97.9) and is the first full-tensor row that is clearly slower to saturate. Dropping $\mathbf{g}^{\text{out}}$@t is almost free (99.9). So Δ is useful polish; ξ_i and fused $\mathbf{g}^{\text{out}}$@t are optional once the state is a tensor.

The second block isolates the input mix. ConcOnlyC alone already averages 98.5. ConcOnly, the same mix without the channel convolution, falls to 79.3. OneLinear, a dense fused projection, is strong at 1024 and 2048 then collapses at 4096 (54.2) despite more parameters (275k vs 175k). ConcOnlyC is not a free lunch on short context; it is the mix that keeps the tensor usable as length grows.

The third block repeats those input mixes with a matrix state, $d_k \times d_v = 16 \times 32$ per head instead of $d_k \times d_r \times d_v = 16 \times 16 \times 32$. All three sit between 21 and 31 average F1. Switching ConcOnlyC, ConcOnly, or OneLinear does not close the gap. The jump from matrix to tensor is the effect; the rest of the recipe is how we feed that tensor.

3.4 Real time-series benchmarks

We evaluate the same $d_{\text{model}} = 128$, depth-2 stack on three real-signal blocks: UEA multivariate classification (Table 3), Informer-style long-horizon forecasting with lookback 96 and horizons

Table 3: UEA multivariate classification accuracy in %. FaceDetection, Heartbeat, and PEMS-SF. Seed 1; learning rate from the Heartbeat probe.

Method	FaceDetection	Heartbeat	PEMS-SF	Avg ↑
Attention	53.4	**72.7**	54.9	60.3
GLA	53.1	67.3	63.0	61.1
Mamba-2	**55.1**	57.6	61.8	58.2
Gated ΔNet	55.4	62.4	61.8	59.9
ΔNet	54.7	62.9	71.7	63.1
RT	54.5	50.7	63.0	56.1
RT/4	54.9	57.1	71.1	61.0
RT/4-Lin	54.9	62.4	**75.7**	**64.4**

Table 4: Multivariate long-term forecasting MSE and MAE at lookback 96 and horizons 96, 192, 336, and 720. Seed 1. A hyphen marks a missing finished run; traffic at horizon 720 OOM'd at batch size 16.

Method		Electricity				Traffic				Weather				Avg ↓
		96	192	336	720	96	192	336	720	96	192	336	720	
Attention	MSE	0.900	0.913	0.907	0.909	1.436	1.437	1.457	-	0.452	0.465	0.488	0.638	0.909
	MAE	0.784	0.781	0.781	0.784	0.803	0.798	0.805	-	0.465	0.474	0.480	0.558	0.683
GLA	MSE	0.312	0.330	0.331	0.335	0.707	0.751	0.739	-	0.317	0.363	0.442	0.538	0.470
	MAE	0.404	0.422	0.422	0.417	0.402	0.428	0.420	-	0.370	0.408	0.455	0.508	0.423
Mamba-2	MSE	0.285	0.313	0.321	-	0.727	0.699	0.712	-	0.290	0.381	0.437	0.540	0.471
	MAE	0.384	0.408	0.414	-	0.416	0.400	0.407	-	**0.358**	0.420	0.451	0.510	0.417
Gated ΔNet	MSE	0.293	0.330	**0.305**	0.318	**0.689**	-	0.744	-	**0.288**	0.413	**0.401**	0.571	**0.435**
	MAE	0.387	0.416	**0.397**	0.401	**0.390**	-	0.419	-	0.359	0.449	0.439	0.519	0.418
ΔNet	MSE	**0.278**	**0.285**	0.320	**0.306**	0.702	0.708	**0.705**	-	0.332	0.374	0.416	0.519	0.449
	MAE	**0.377**	**0.381**	0.412	**0.396**	0.402	0.402	**0.399**	-	0.374	0.412	0.442	0.493	**0.408**
RT	MSE	0.308	0.297	0.328	0.325	0.698	0.715	0.706	-	0.303	0.396	0.434	0.530	0.458
	MAE	0.400	0.394	0.418	0.408	0.401	0.410	0.405	-	0.370	0.435	0.456	0.503	0.418
RT/4	MSE	0.309	0.334	0.313	0.336	0.712	0.738	0.716	-	0.300	0.420	0.453	0.511	0.467
	MAE	0.400	0.425	0.408	0.419	0.408	0.432	0.409	-	0.369	0.450	0.464	0.489	0.425
RT/4-Lin	MSE	0.289	0.318	0.314	0.316	0.702	**0.682**	0.720	-	0.328	**0.360**	0.408	**0.492**	0.448
	MAE	0.386	0.411	0.407	0.402	0.398	**0.388**	0.408	-	0.376	**0.405**	**0.433**	**0.478**	**0.408**

$\{96, 192, 336, 720\}$ (Table 4), and Google Speech Commands 10-way classification on 16 kHz raw audio (∼16k steps; Table 5). Learning rates come from a Heartbeat probe over a small grid; 10^{-3} is selected for every architecture and then held fixed across datasets (seed 1).

Full RunningTensor uses per-head state $16 \times 16 \times 32$, matching the GDN layout at this width ($d_k = d_r = 16$, $d_v = 32$). That is 8192 slots per head, versus 512 for the matrix 16×32 used in linear attention and in the matrix ablations, or 256 if one compares to a square 16×16 map. The cubic state is why full RT does not finish SC10 under the shared 16 h budget. RT/4 keeps the value axis and shrinks the two contracted axes by 4, so $d_k = d_r = 8$ and $d_v = 32$. The state is $8 \times 8 \times 32 = 2048$ slots per head: a quarter of full RT, still $4\times$ a 16×32 matrix and $8\times$ a 16×16 matrix. RT/4-Lin uses that same reduced tensor, but replaces ConcOnlyC by the default linear input projection plus short convolution. We report both on the real-signal grid, and only those two on SC10.

On UEA (Table 3), FaceDetection is flat across methods (∼53 to 55%); the only spread is on Heartbeat and PEMS-SF. RT/4-Lin leads PEMS-SF (75.7%); full RT is worst on Heartbeat (50.7%), likely a failed optimization cell rather than a model-class effect. On forecasting (Table 4), softmax attention is unusable (∼0.91 MSE); GLA, Mamba-2, ΔNet, and RT variants cluster near ∼0.43 to 0.47 MSE on completed cells. Traffic at horizon 720 is missing for every method (OOM at batch size 16). Speech Commands (Table 5) is the only block where length strongly separates architectures: attention and GLA collapse (∼20 to 27%), ΔNet reaches 82.0%, and RT/4 variants sit just behind at 77 to 78%.

Table 5: Speech Commands 10-way classification accuracy in % on 16 kHz raw audio, about 16k steps. Seed 1. Full RT omitted; it does not finish under the shared budget.

Method	**SC10** ↑
Attention	20.0
GLA	27.3
Mamba-2	73.8
Gated ΔNet	51.7
ΔNet	**82.0**
RT	-
RT/4	77.8
RT/4-Lin	77.5

4 Related Work

Linear Attention and Fast Weight Mechanisms. A prominent direction for improving sequence modeling efficiency replaces quadratic self-attention with linear-time alternatives. Linformer reduces complexity by projecting keys and values into a fixed-dimensional space, while Linear Transformers reformulate attention using kernel-based similarity functions to enable associative computation Katharopoulos et al. (2020); Wang et al. (2020). Performer further approximates softmax attention with random feature methods Choromanski et al. (2021). Later models enhance stability and expressivity through architectural modifications: RetNet introduces decay and rotational structure, and Gated Linear Attention incorporates learnable gating Sun et al. (2023); Yang et al. (2024b). These approaches can be interpreted as instances of fast weight programmers, where weights are dynamically updated via input-dependent outer products, enabling implicit online adaptation during inference Schlag et al. (2021). This perspective connects to earlier work on fast weight memory and to recent findings that Transformers can perform implicit optimization while processing sequences Schmidhuber (1992; 1993); von Oswald et al. (2023).

State Space Models and Gated Linear RNNs. State space models (SSMs) provide an alternative framework for efficient sequence modeling by structuring recurrence in a way that supports parallel computation. Early formulations rely on fixed or structured transition matrices that allow convolutional evaluation Gu et al. (2022); Li et al. (2023). Subsequent models, including DSS, GSS, S5, H3, and Mamba, improve expressivity through diagonal, low-rank, or selective state transitions Fu et al. (2023); Gu & Dao (2024); Gupta et al. (2022); Mehta et al. (2023); Smith et al. (2023). Closely related are linear recurrent architectures such as LRU, HGRN, and RWKV, which share similar computational principles Orvieto et al. (2023); Peng et al. (2023); Qin et al. (2023). A key evolution in this space is the move from data-independent transitions to input-dependent gating mechanisms, inspired by classical gated RNNs but redesigned to remove dependence on previous hidden states, thereby preserving parallelism Gers et al. (2000); Greff et al. (2015); Martin & Cundy (2018). This gating paradigm, also termed selective state updates, has become central to modern architectures such as Mamba and its successors.

Delta Rule, Online Learning, and Hybrid Architectures. Another unifying perspective frames these models as performing online learning through weight updates. The delta rule, used in DeltaNet, provides greater memory capacity than Hebbian-style updates and leads to flexible identity-plus-low-rank transition structures Gardner (1988); Irie et al. (2021); Yang et al. (2024c). These properties support improved reasoning and state tracking but also introduce stability and scalability challenges. Recent work addresses these limitations through parallelization strategies, normalization, and structured approximations Sun et al. (2025); Yang et al. (2024c). Extensions incorporating nonlinear objectives or richer transition dynamics further enhance expressivity, though often at the cost of reduced parallel efficiency Behrouz et al. (2026); Grazzi et al. (2025); Siems et al. (2025). Finally, hybrid architectures that combine attention with linear recurrent or SSM layers, either across or within layers, have emerged as a practical way to balance efficiency and performance Hua et al. (2022); Ren et al. (2025); Waleffe et al. (2024).

5 CONCLUSION

RunningTensor lifts the recurrent state of linear attention from a matrix to an order-3 tensor, while keeping a rank-1 update and a readout that is still linear in sequence length. On MQAR at width 128 and lengths 1024 to 4096, a harder band than the usual width-512 / length-512 setting, the full tensor matches attention at 100 F1 with about $5\times$ fewer non-embedding parameters per layer. Matrix-state copies of the same recipe stay in the 20s. That gap is the main empirical claim: extra modes in S_t buy associative capacity that gating and input mix cannot recover on their own.

On real signals the same extra state still shows up where length and retrieval matter. RT/4-Lin leads the UEA average, and on Informer-style forecasting the linear models, including both RT variants, sit in a tight band well ahead of softmax attention. The longest test is Speech Commands at $\sim$16k steps: attention and GLA collapse, ΔNet is strongest, and RT/4 and RT/4-Lin land just behind it.

AI USE STATEMENT

In this work, we used generative AI tools to design or provide feedback on research methodology and experiments, to implement methods, to support qualitative and thematic data analysis, and to interpret results. We have reviewed all AI-assisted work and take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

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. In *International Conference on Learning Representations (ICLR)*, 2024.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. volume 38, pp. 113506–113543, 2026.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans (eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.

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

Daniel Y. Fu, Tri Dao, Khaled K. Saab, Armin W. Thomas, Atri Rudra, and Christopher Ré. Hungry hungry hippos: Towards language modeling with state space models. In *International Conference on Learning Representations (ICLR)*, 2023.

Elizabeth Gardner. The space of interactions in neural network models. *Journal of Physics A: Mathematical and General*, 21(1):257–270, 1988.

Felix A. Gers, Jürgen Schmidhuber, and Fred A. Cummins. Learning to forget: Continual prediction with LSTM. *Neural Computation*, 12(10):2451–2471, 2000.

Riccardo Grazzi, Julien N. Siems, Jörg K. H. Franke, Arber Zela, Frank Hutter, and Massimiliano Pontil. Unlocking state-tracking in linear RNNs through negative eigenvalues. In *International Conference on Learning Representations (ICLR)*, 2025.

Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232, 2015.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *Conference on Language Modeling (COLM)*, 2024.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations (ICLR)*, 2022.

Ankit Gupta, Albert Gu, and Jonathan Berant. Diagonal state spaces are as effective as structured state spaces. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.

Weizhe Hua, Zihang Dai, Hanxiao Liu, and Quoc V. Le. Transformer quality in linear time. In *International Conference on Machine Learning (ICML)*, 2022.

Kazuki Irie, Imanol Schlag, Róbert Csordás, and Jürgen Schmidhuber. Going beyond linear transformers with recurrent fast weight programmers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning (ICML)*, pp. 5156–5165, 2020.

Yuhong Li, Tianle Cai, Yi Zhang, Deming Chen, and Debadeepta Dey. What makes convolutional models great on long sequence modeling? In *International Conference on Learning Representations (ICLR)*, 2023.

Bo Liu, Rui Wang, Lemeng Wu, Yihao Feng, Peter Stone, and Qiang Liu. Longhorn: State space models are amortized online learners. In *International Conference on Learning Representations (ICLR)*, 2025.

Eric Martin and Chris Cundy. Parallelizing linear recurrent neural nets over sequence length. In *International Conference on Learning Representations (ICLR)*, 2018.

Harsh Mehta, Ankit Gupta, Ashok Cutkosky, and Behnam Neyshabur. Long range language modeling via gated state spaces. In *International Conference on Learning Representations (ICLR)*, 2023.

Antonio Orvieto, Samuel L. Smith, Albert Gu, Anushan Fernando, Çağlar Gülçehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. In *International Conference on Machine Learning (ICML)*, 2023.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Leon Derczynski, Xingjian Du, Matteo Grella, Kranthi Gv, Xuzheng He, Haowen Hou, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartłomiej Koptyra, Hayden Lau, Jiaju Lin, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Guangyu Song, Xiangru Tang, Johan Wind, Stanisław Woźniak, Zhenyuan Zhang, Qinghua Zhou, Jian Zhu, and Rui-Jie Zhu. RWKV: Reinventing RNNs for the Transformer Era. In *Findings of the Association for Computational Linguistics: EMNLP*, 2023.

Zhen Qin, Songlin Yang, and Yiran Zhong. Hierarchically gated recurrent networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. OpenAI technical report, 2018.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. OpenAI technical report, 2019.

Liliang Ren, Yang Liu, Yadong Lu, Yelong Shen, Chen Liang, and Weizhu Chen. Samba: Simple hybrid state space models for efficient unlimited context language modeling. In *International Conference on Learning Representations (ICLR)*, 2025.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning (ICML)*, 2021.

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

Jürgen Schmidhuber. Reducing the ratio between learning complexity and number of time varying variables in fully recurrent nets. In *International Conference on Artificial Neural Networks (ICANN)*, 1993.

Julien Siems, Timur Carstensen, Arber Zela, Frank Hutter, Massimiliano Pontil, and Riccardo Grazzi. Deltaproduct: Improving state-tracking in linear RNNs via householder products. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Jimmy T. H. Smith, Andrew Warrington, and Scott W. Linderman. Simplified state space layers for sequence modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

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. In *International Conference on Machine Learning (ICML)*, 2025.

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, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

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

Johannes von Oswald, Maximilian Schlegel, Alexander Meulemans, Seijin Kobayashi, Eyvind Niklasson, Nicolas Zucchet, Nino Scherrer, Nolan Miller, Mark Sandler, Blaise Agüera y Arcas, Max Vladymyrov, Razvan Pascanu, and João Sacramento. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning (ICML)*, 2023.

Roger Waleffe, Wonmin Byeon, Duncan Riach, Brandon Norick, Vijay Korthikanti, Tri Dao, Albert Gu, Ali Hatamizadeh, Sudhakar Singh, Deepak Narayanan, Garvit Kulshreshtha, Vartika Singh, Jared Casper, Jan Kautz, Mohammad Shoeybi, and Bryan Catanzaro. An empirical study of mamba-based language models. *arXiv preprint arXiv:2406.07887*, 2024.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024a.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

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 (ICML)*, 2024b.

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 (NeurIPS)*, 2024c.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated Delta Networks: Improving Mamba2 with Delta Rule. In *International Conference on Learning Representations (ICLR)*, 2025b.