SPECTRAL POLYNOMIAL LINEAR ATTENTION (SPLA): POLYNOMIAL SINGULAR-VALUE MAPS FOR GATED LINEAR MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Linear models have gained widespread attention as a viable alternative to reduce the cost of full-attention Transformers. However, due to their fixed-size state space, the spectral behavior of the state space directly determines their performance. Recent work has proposed that their dominant components cause the effective rank of the state to collapse, making it difficult to maintain long-range memory capacity. We draw on the ideas of the Muon optimizer and focus on recent work such as MesaNet, finding that it essentially regulates the key-value matrix of all past states. We argue that a broader singular value spectrum of the keys and values may provide the model with more choices. Based on this, we propose Spectral Polynomial Linear Attention (SPLA). The architecture we propose provides an alternative way of using the state of linear models through high-order polynomial approximation function transformations. We further conduct experiments on different tasks, and the results show that our method effectively preserves the long-term recall capacity of linear models while maintaining good inference speed.

1 INTRODUCTION

The softmax attention mechanism Vaswani et al. (2017) has demonstrated remarkable success across diverse domains such as natural language processing Radford et al. (2019); Brown et al. (2020); Touvron et al. (2023a) and computer vision Dosovitskiy et al. (2020), largely due to its strong parallelizability, which enables effective data utilization and captures long-range temporal dependencies. However, its primary limitation lies in the linearly growing memory footprint and inference latency with increasing input sequence length. To address this issue, linear attention Katharopoulos et al. (2020); Qin et al. (2024); Yang et al. (2024a); Chou et al. (2024); Beck et al. (2024) and state-space models (SSMs) Gu et al. (2022a;b); Gu & Dao (2024) have been proposed as efficient alternatives. By eliminating the nonlinear computation between queries and keys, these approaches allow pre-computation of key–value (KV) states and maintain a fixed-size hidden state that propagates information along the temporal dimension, thereby achieving constant memory consumption and inference latency regardless of sequence length Merrill et al. (2024); Grazzi et al. (2025). This enables linear attention to achieve faster inference speed and lower computational overhead compared to full-attention Transformers in ultra-long sequence inference, and it has been incorporated into hybrid model architectures and widely adopted Qwen (2025); Chen et al. (2026).

A critical challenge for such linear models, however, is the inherent constraint imposed by the fixed state size Jelassi et al. (2024); Du et al. (2026), which makes linear models inevitably suffer from memory loss under long-sequence memory. A number of works have shown that single-layer linear attention modules are prone to being dominated by a few principal directions, leading to a progressive reduction in the effective information content of the state matrix over time Pan et al. (2026); Nguyen et al. (2026) and leading to confusion in long-sequence memory.

A series of works Liu et al. (2024); Yang et al. (2025a); Sun et al. (2025); Behrouz et al. (2026) have pointed out that linear models essentially implement a form of test-time behavior at runtime, considering linear attention from the perspective of loss functions and optimization Irie et al. (2022); Wang et al. (2025), while naturally introducing considerations from optimization research. How to more

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.

effectively perform token-wise step-by-step optimization of the states used by linear models directly determines their long-term performance capability. Therefore, several studies have attempted to remedy this issue by constructing global gradient losses or incorporating auxiliary states to refine the memory used for retrieval, aiming to preserve long-term memorization capacity von Oswald et al. (2025); Peng et al. (2026); Shaj et al. (2026), effectively improving the long-sequence memory capability of linear models.

In this work, we first analyze the core principles of these existing methods and observe that they can largely be characterized as decomposing the eigenspace of the key vector matrix and ultimately reducing to operations on its singular values. Through singular value operations, the vector directions occupying the dominant components are effectively shrunk, which effectively enhances the relative contribution of the remaining components and thus achieves a certain whitening effect. Inspired by this insight, we consider how to perform transformations at the level of matrix singular values. We draw on the Muon Jordan et al. (2024) optimization framework and propose approximating various singular value transformation functions through polynomial replacement, enabling the model to autonomously choose how to operate on the vector components of the state space. We derive an efficient parallel computation algorithm for the proposed method and implement the corresponding kernel operator. Finally, we conduct extensive experiments on a series of tasks to evaluate its effectiveness.

2 Background and Related Works

2.1 Notation

For simplicity, we assume that the query, key, and value vectors use the same head dimension d. We use plain-text letters (e.g., x, y) for scalars, bold lowercase italic letters $\boldsymbol{q}_t, \boldsymbol{k}_t \in \mathbb{R}^{1\times d}$ to denote the t-th row vector, and use bold uppercase italic letters $\boldsymbol{S}_t \in \mathbb{R}^{d\times d}$ to denote the t-th matrix. We use $\boldsymbol{Q}_{[i]} := \boldsymbol{q}_{ic+1:(i+1)c} \in \mathbb{R}^{c\times d}$, and use $\boldsymbol{S}_{[i]} := \boldsymbol{S}_{ic} \in \mathbb{R}^{d\times d}$ to present the i-th chunk query matrix and initial chunk state for chunk-wise parallel notation. We use $\boldsymbol{q}^t_{[i]} = \boldsymbol{q}_{ic+t} \in \mathbb{R}^{1\times d}$, and use $\boldsymbol{S}^t_{[i]} := \boldsymbol{S}_{ic+t} \in \mathbb{R}^{d\times d}$ to present the t-th query vector and current state of i-th chunk.

2.2 Softmax and linear attention

At each time step t, the self-attention mechanism linearly projects the current input x_t into query $\boldsymbol{q}_t$, key $\boldsymbol{k}_t$,value $\boldsymbol{v}_t$. Subsequently, the query $\boldsymbol{q}_t$ at the current step retrieves relevance scores against all historical time steps via the softmax function. After obtaining the normalized attention weights through the softmax normalization factor, the output is computed as a weighted sum over the values $\boldsymbol{v}_t$, formulated as follows:

$$\boldsymbol{o}_t = \frac{\sum_{i=1}^{t} exp(\frac{\boldsymbol{q}_t \boldsymbol{k}_i^\top}{\sqrt{d}})\boldsymbol{v}_i}{\sum_{i=1}^{t} exp(\frac{\boldsymbol{q}_t \boldsymbol{k}_i^\top}{\sqrt{d}})}. \tag{1}$$

Linear attention combined with state-space models (SSMs) eliminates the nonlinear softmax operator Schlag et al. (2021); Qin et al. (2023), allowing the computation in equation 1 to be reused along the temporal dimension by retaining the precomputed results between keys and values. This enables constant memory and time complexity during inference, independent of the sequence length. The core challenge of linear attention lies in two aspects: how to effectively utilize a fixed memory budget, and how to construct the state transition matrix between consecutive time steps.

From the perspective of online optimization, existing linear attention methods primarily focus on two objectives Liu et al. (2024); Yang et al. (2025a): minimizing the discrepancy between adjacent time steps, and reducing the reconstruction error when querying the current state with the retrieval vector $\boldsymbol{k}_t$ against the target vector $\boldsymbol{v}_t$. Formally, this can be expressed as:

$$\boldsymbol{S}_t = \underset{\boldsymbol{S}_t \in \mathbb{R}^{d\times d}}{\arg\min} \|\boldsymbol{k}_t \boldsymbol{S}_t - \boldsymbol{v}_t\|^2 + \lambda \|\boldsymbol{S}_t - \boldsymbol{S}_{t-1}\|_F^2, \tag{2}$$

where the first term enforces retrieval fidelity, and the second term regularizes the state update to preserve continuity across time, leading to longhorn Liu et al. (2024) and DeltaNet Yang et al.

(2024b). The state transition formula is:

$$\boldsymbol{S}_t = (\boldsymbol{I} - \epsilon \boldsymbol{k}_t^\top \boldsymbol{k}_t)\boldsymbol{S}_{t-1} + \epsilon \boldsymbol{k}_t^\top \boldsymbol{v}_t, \epsilon = \frac{1}{\lambda + \boldsymbol{k}_t \boldsymbol{k}_t^\top}. \tag{3}$$

The aforementioned approaches rely solely on single-step state updates. In contrast, Mesalayer Von Oswald et al. (2023), MesaNet von Oswald et al. (2025) and the Gated KalmaNet (GKA) Peng et al. (2026) framework adopt a global optimization perspective, where the optimization objective at each step is formulated as the cumulative sum over all past time steps, while simultaneously imposing a norm constraint on the state at each time step,

$$\boldsymbol{S}_t = \underset{\boldsymbol{S}_t \in \mathbb{R}^{d \times d}}{\arg\min} \sum_{i=1}^{t} \beta_i \left\| \boldsymbol{k}_i \boldsymbol{S}_t - \boldsymbol{v}_i \right\|^2 + \lambda \|\boldsymbol{S}_t\|_F^2, \tag{4}$$

which leads to the following update form for the state:

$$\boldsymbol{o}_t = \boldsymbol{q}_t (\lambda \boldsymbol{I} + \sum_{i=1}^{t} \beta_i \boldsymbol{k}_i^\top \boldsymbol{k}_i)^{-1} (\sum_{i=1}^{t} \beta_i \boldsymbol{k}_i^\top \boldsymbol{v}_i). \tag{5}$$

When using decay factor γ for state $\boldsymbol{H}$ like GLA Yang et al. (2024a), we can construct a mapping $\boldsymbol{k} \mapsto \boldsymbol{k}', \boldsymbol{v} \mapsto \boldsymbol{v}'$ which applies a cumulative decay factor to each past key up to the current time step, i.e., $\boldsymbol{k}_i' = (\prod_{j=i+1}^{t} \sqrt{\gamma_j})\sqrt{\beta_i}\boldsymbol{k}_i, \boldsymbol{v}_i' = (\prod_{j=i+1}^{t} \sqrt{\gamma_j})\sqrt{\beta_i}\boldsymbol{v}_i$. This formulation allows us to construct a transformed key matrix $\boldsymbol{K}_t', \boldsymbol{V}_t'$ at each time step effectively reweighting historical keys according to their temporal distance, then equation 5 can be written as:

$$\boldsymbol{o}_t = \boldsymbol{q}_t (\lambda \boldsymbol{I} + \boldsymbol{K}_t'^\top \boldsymbol{K}_t')^{-1} (\boldsymbol{K}_t'^\top \boldsymbol{V}_t'). \tag{6}$$

2.3 MUON AND SVD

Muon Jordan et al. (2024) is a recently proposed optimizer that applies the msign operation to the momentum of the update states, which flattens the singular values of the update momentum to 1 and thereby synchronizes the update magnitudes across all directions. This property invites interpretations from two perspectives. First, it characterizes the optimizer state in terms of the singular values of a matrix, which naturally aligns with linear attention models that maintain two-dimensional (matrix-valued) states, inspiring follow-up works such as MuonSSM Nguyen et al. (2026). Second, it demonstrates that the singular values of a matrix can be directly manipulated without explicitly computing the SVD Su (2026): Muon employs the Newton–Schulz iteration to push the singular values of an arbitrary matrix toward 1.

Motivated by Muon, we observe that equation 6 in fact performs a singular value scaling transformation on the transformed key matrix $\boldsymbol{K}_t'$. That is, suppose the transformed key matrix $\boldsymbol{K}_t'$ is decomposed via singular value decomposition (SVD) as $\boldsymbol{K}_t' = \boldsymbol{U}_t \boldsymbol{\Sigma}_t \boldsymbol{V}_t^\top$, where $\boldsymbol{U}_t, \boldsymbol{V}_t$ are orthogonal matrices, and $\boldsymbol{\Sigma}_t$ contains the singular values in descending order. Compared to the scalar Gated Linear Attention(GLA) Yang et al. (2024a) or RetNet Sun et al. (2023) algorithm, which directly computes $\boldsymbol{q}_t \boldsymbol{K}_t'^\top \boldsymbol{V}_t'$, these methods shrink the singular values of the original $\boldsymbol{K}_t'$ to the new matrix $\boldsymbol{K}_{st}'$ using the following function,

$$\psi(x) = \frac{x}{\lambda + x^2}, \tag{7}$$

then compute $\boldsymbol{q}_t \boldsymbol{K}_{st}'^\top \boldsymbol{V}_t'$, which effectively suppresses the dominant singular vector components that tend to dominate the state, thereby serving as a whitening operation. This rebalances the contribution across different singular directions, amplifying the relative proportion of less prominent yet potentially informative historical components in the state representation.

3 METHOD

From the perspective of the singular value spectrum, MesaNet and a series of prior works such as GKA share a common limitation: the update to the matrix state K' exhibits non-monotonic behavior. Although this non-monotonic compression mitigates the dominant directions of the matrix

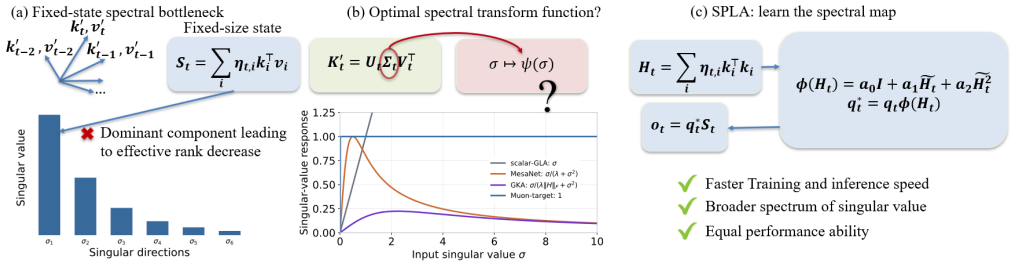

Figure 1: Overview of SPLA. (a) The inherent bottleneck of fixed state size, where the matrix state is influenced by the dominant direction. (b) The existing models essentially vary the singular value function of the $\boldsymbol{K}$-matrix. (c) The method and advantages of SPLA.

to some extent, it remains difficult to determine whether the current spectral transformation function is optimal as Figure 1(b). Meanwhile, we argue that different layers and attention heads of the model may require different singular value transformation functions. Motivated by these observations, our method assumes preconditioning the query with $(\lambda \boldsymbol{I} + \boldsymbol{K}'^{\top}\boldsymbol{K}')^{-\theta}$. The map is applied to a gated running Gram matrix of keys, uses only matrix–vector products with that Gram matrix, and admits a chunked forward scan together with an exact reverse-mode backward pass.

3.1 Gated Gram State

Let $\boldsymbol{q}_t, \boldsymbol{k}_t, \boldsymbol{v}_t \in \mathbb{R}^{1\times d}$ denote the query, key, and value at time t, and let $\gamma_t \in [0,1]$ be the forget gate. We maintain a symmetric Gram matrix $\boldsymbol{H}_t \in \mathbb{R}^{d\times d}$ by the gated rank-one recurrence:

$$\boldsymbol{H}_t = \gamma_t \boldsymbol{H}_{t-1} + \boldsymbol{k}_t^{\top}\boldsymbol{k}_t, \qquad \boldsymbol{H}_0 = \boldsymbol{0}. \tag{8}$$

Here we ignore the β_t because it can be absorbed into $\boldsymbol{k}_t, \boldsymbol{v}_t$. Unrolling equation 8 gives $\boldsymbol{H}_t = \sum_{i=1}^{t} \eta_{t,i}\, \boldsymbol{k}_i^{\top}\boldsymbol{k}_i$ with $\eta_{t,i} = \prod_{j=i+1}^{t} \gamma_j$. Thus $\boldsymbol{H}_t$ is a fading-memory analogue of $\boldsymbol{K}^{\top}\boldsymbol{K}$ on the prefix $(\boldsymbol{k}_1, \ldots, \boldsymbol{k}_t)$. Ridge-style linear attention would apply $(\boldsymbol{H}_t + \lambda \boldsymbol{I})^{-1}$ to $\boldsymbol{q}_t$. The associated singular-value map $\sigma \mapsto \sigma/(\sigma^2 + \lambda)$ of $\boldsymbol{K}'$ is non-monotone. There is no clear indication that this singular value transformation function exhibits optimal performance. Therefore, we consider more arbitrary transformation functions:

$$\boldsymbol{q}_t^{*} = \boldsymbol{q}_t \phi(\boldsymbol{H}_t)\,, \qquad \phi(\boldsymbol{H}_t) = (\lambda \boldsymbol{I} + \boldsymbol{H}_t)^{-\theta}. \tag{9}$$

3.2 Three-term expansion of $(\lambda + X)^{-\theta}$

We approximate the matrix power using a Muon-like matrix polynomial method. To reconcile computational efficiency with the feasibility of backpropagation, we employ a single iteration of a high-order polynomial approximation, and introduce learnable coefficients that allow the model to adaptively select different singular value transformation behaviors. We normalize the input $\boldsymbol{H}_t$ by the Frobenius norm $f_t \in \mathbb{R}$ with a small $\varepsilon > 0$, so that its singular value falls within the range of $[0, 1]$, ensure the convergence of expansion:

$$f_t = \|\boldsymbol{H}_t\|_F + \varepsilon \qquad \widetilde{\boldsymbol{H}}_t = \boldsymbol{H}_t / f_t, \tag{10}$$

After that, use Taylor expansion:

$$(\lambda + X)^{-\theta} = a_0 + a_1 X + a_2 X^2 + O(\|X\|^3), \tag{11}$$

$$\phi(\boldsymbol{H}_t) = a_0\, I + a_1\, \widetilde{\boldsymbol{H}}_t + a_2\, \widetilde{\boldsymbol{H}}_t^2, \tag{12}$$

The triple (a_0, a_1, a_2) may be frozen at these Taylor values or treated as per-head learnable scalars. Acting on the query yields the vector recurrence used in the implementation,

$$\begin{aligned} \boldsymbol{o}_t^{(1)} &= \boldsymbol{q}_t \boldsymbol{H}_t / f_t, \\ \boldsymbol{o}_t^{(2)} &= \boldsymbol{o}_t^{(1)} \boldsymbol{H}_t / f_t = \boldsymbol{q}_t \boldsymbol{H}_t^2 / f_t^2, \\ \boldsymbol{q}_t^{*} &= a_0\, \boldsymbol{q}_t + a_1\, \boldsymbol{o}_t^{(1)} + a_2\, \boldsymbol{o}_t^{(2)}. \end{aligned} \tag{13}$$

3.3 FORWARD COMPUTATION

Forward Recurrent. Equation 8 is linear in $\boldsymbol{H}_t$ and is evaluated with a chunked scan. Partition the sequence into chunks of length c. Inside a chunk starting at time s for the local cumulative gate and let $\boldsymbol{H}_0$ be the incoming chunk state. Then

$$\boldsymbol{H}_t = \eta_{t,0}\boldsymbol{H}_0 + \sum_{j=1}^{t} \eta_{t,j}\boldsymbol{k}_j^\top \boldsymbol{k}_j. \tag{14}$$

Applying $\boldsymbol{H}_t$ to a vector without materializing a $d \times d$ matrix per token uses the same identity as gated linear attention:

$$\boldsymbol{q}_t\boldsymbol{H}_t = \eta_{t,0}\, \boldsymbol{q}_t\boldsymbol{H}_0 + \sum_{j=1}^{t} \eta_{t,j}\, (\boldsymbol{q}_t\boldsymbol{k}_j^\top)\, \boldsymbol{k}_j. \tag{15}$$

The second-order term $\boldsymbol{o}_t^{(2)}$ is obtained by repeating equation 15 with $\boldsymbol{o}_t^{(1)}$ in place of $\boldsymbol{q}_t$. The scale f_t is accumulated from the same chunk quantities Peng et al. (2026),

$$\|\boldsymbol{H}_t\|_F^2 = \eta_{t,0}^2\|\boldsymbol{H}_0\|_F^2 + 2\eta_{t,0}\sum_{j=1}^{t}\eta_{t,j}\boldsymbol{k}_j\boldsymbol{H}_0\boldsymbol{k}_j^\top + \|\sum_{j=1}^{t}\eta_{t,j}\,\boldsymbol{k}_j^\top\boldsymbol{k}_j\|^2. \tag{16}$$

The preconditioned query is mixed with the original query and consumed by a gated linear attention (GLA) readout,

$$\tilde{\boldsymbol{q}}_t = (1-\alpha_t)\,\boldsymbol{q}_t + \alpha_t\,\boldsymbol{q}_t^*, \qquad \boldsymbol{y}_t = \mathrm{GLA}(\tilde{\boldsymbol{q}}_t, \boldsymbol{k}_t, \boldsymbol{v}_t, g_t^{(kv)}), \tag{17}$$

where $\alpha_t \in (0,1)$ is a learned interpolation gate. When $\alpha_t = 0$ the layer reduces to ordinary GLA; when $\alpha_t = 1$ the query is fully preconditioned by $\phi(\boldsymbol{H}_t)$.

Forward Chunk-wise Parallel. The chunk-wise parallel computation Hua et al. (2022); Sun et al. (2023); Yang et al. (2024a;b) is implemented as before. Specifically, we first reuse the state update function of GLA to obtain the state $\boldsymbol{H}_{[i]}$ of each chunk, and then compute the Frobenius norm $\|\boldsymbol{H}_t\|_F$ at each time step in parallel. Let $A_{[i]j,k} = \eta_{ic+j,ic+k}$ if $j \geq k$ else 0, $\boldsymbol{\eta}_{[i]} = [\eta_{ic+1,ic}, \cdots, \eta_{ic+c,ic}]$ then the equation 15 can be computed parallel by this operator:

$$\mathcal{T}(\boldsymbol{Z}_{[i]}) = (\boldsymbol{\eta}_{[i]}^\top \mathbf{1}_d) \odot (\boldsymbol{Z}_{[i]}\boldsymbol{H}_{[i]}) + \big(\boldsymbol{A}_{[i]} \odot (\boldsymbol{Z}_{[i]}\boldsymbol{K}_{[i]}^\top)\big)\boldsymbol{K}_{[i]}, \tag{18}$$

The equation 16 can be computed in parallel by three term, while the third with row-sum$((\boldsymbol{A}(\boldsymbol{H}_K \odot \boldsymbol{H}_K)) \odot \boldsymbol{A})$, where $\boldsymbol{H}_K = \boldsymbol{K}\boldsymbol{K}^\top$. The other two can be computed easily, then we get the chunk-wise Frobenius norm $\boldsymbol{f}_{[i]} \in \mathbb{R}^{1\times c}$. We put the total computation in appendix A.

Based on these quantities, we compute $\boldsymbol{O}^{(1)}$ from $\boldsymbol{Q}$ by operator $\mathcal{T}(\cdot)$ and subsequently obtain $\boldsymbol{O}^{(2)}$ from $\boldsymbol{O}^{(1)}$, yielding the final preconditioned query $\boldsymbol{Q}^*$:

$$\boldsymbol{Q}_{[i]}^* = a_0\boldsymbol{Q}_{[i]} + a_1\mathcal{T}\big((\mathrm{diag}(\boldsymbol{f}_{[i]})^{-1})\boldsymbol{Q}_{[i]}\big) + a_2\mathcal{T}\big(\mathcal{T}((\mathrm{diag}(\boldsymbol{f}_{[i]})^{-2})\boldsymbol{Q}_{[i]}\big) \tag{19}$$

3.4 BACKWARD PROPAGATION

Backward recurrent. Write $\nabla\boldsymbol{q}_t^* = \partial\mathcal{L}/\partial\boldsymbol{q}_t^*$ for the upstream gradient of equation 13 and $\nabla\boldsymbol{H}_t = \partial\mathcal{L}/\partial\boldsymbol{H}_t$ for the gradient of the Gram state, including every later use of $\boldsymbol{H}_t$ through equation 8. Inner products of matrices are Frobenius, $\langle\boldsymbol{A},\boldsymbol{B}\rangle_F = \mathrm{tr}(\boldsymbol{A}^\top\boldsymbol{B})$. The GLA value gradient $\nabla\boldsymbol{v}_t$ is separate: it comes only from the readout in equation 17 and does not depend on $\{\nabla\boldsymbol{H}_t\}$.

We then record how $\nabla\boldsymbol{q}_t, \nabla\boldsymbol{k}_t, \nabla\gamma_t$ couple to the whole family $\{\nabla\boldsymbol{H}_t\}$, then show that only chunk-end matrices $\nabla\boldsymbol{H}$ need to be stored, and finally combine the two observations into the actual backward kernels.

The polynomial equation 13 depends on $(\boldsymbol{q}_t, \boldsymbol{H}_t)$ at the same time index only. Because $\phi(\boldsymbol{H}_t)$ is self-adjoint,

$$\nabla\boldsymbol{q}_t = \nabla\boldsymbol{q}_t^*\phi(\boldsymbol{H}_t) \;= a_0\,\nabla\boldsymbol{q}_t^* + a_1\,\nabla\boldsymbol{q}_t^*\,\widetilde{\boldsymbol{H}}_t \;+ a_2\,\nabla\boldsymbol{q}_t^*\,\widetilde{\boldsymbol{H}}_t^2\,. \tag{20}$$

Thus $\nabla \boldsymbol{q}_t$ is a function of the forward state $\boldsymbol{H}_t$ and of $\nabla \boldsymbol{q}_t^*$; it does not depend on $\{\nabla \boldsymbol{H}_t\}$. If (a_0, a_1, a_2) are learned, $\nabla a_0 = \sum_t \langle \nabla \boldsymbol{q}_t^*, \boldsymbol{q}_t \rangle$ and likewise for a_1 against $\boldsymbol{o}_t^{(1)}$ and a_2 against $\boldsymbol{o}_t^{(2)}$.

Differentiating equation 13 at fixed $\boldsymbol{q}_t$ produces a local Gram gradient $\nabla \boldsymbol{H}_t^{\mathrm{loc}}$ (the part of $\nabla \boldsymbol{H}_t$ that does not yet include future tokens). The two applications of $\boldsymbol{H}_t$ and the scalar $f_t = \|\boldsymbol{H}_t\|_F + \varepsilon$ are summarized by three vector/scalar statistics of $\nabla \boldsymbol{q}_t^*$:

$$\begin{aligned} \nabla \boldsymbol{o}_t^{(1)} &= \nabla \boldsymbol{q}_t^* \, \widetilde{\boldsymbol{H}}_t = \mathcal{T}(\nabla \boldsymbol{q}_t^*)/f_t, \\ \boldsymbol{u}_t &= \big(a_1 \, \nabla \boldsymbol{q}_t^* + a_2 \, \nabla \boldsymbol{o}_t^{(1)}\big)/f_t, \\ \boldsymbol{w}_t &= a_2 \, \nabla \boldsymbol{q}_t^*/f_t, \\ \xi_t &= -\frac{a_1 \, \langle \boldsymbol{o}_t^{(1)}, \nabla \boldsymbol{q}_t^* \rangle + 2a_2 \, \langle \boldsymbol{o}_t^{(2)}, \nabla \boldsymbol{q}_t^* \rangle}{f_t^2}. \end{aligned} \tag{21}$$

Here $\boldsymbol{u}_t$ ($\boldsymbol{w}_t$) is the adjoint factor of the first (second) multiply by $\boldsymbol{H}_t$ and ξ_t is the adjoint of the Frobenius scale. They assemble the local Gram gradient as a sum of two rank-one terms plus a multiple of $\boldsymbol{H}_t$ itself:

$$\nabla \boldsymbol{H}_t^{\mathrm{loc}} = \boldsymbol{u}_t^\top \boldsymbol{q}_t + \boldsymbol{w}_t^\top \boldsymbol{o}_t^{(1)} + \xi_t \, \boldsymbol{H}_t. \tag{22}$$

The recurrence equation 8 couples the local pieces, hence:

$$\nabla \boldsymbol{H}_t = \nabla \boldsymbol{H}_t^{\mathrm{loc}} + \gamma_{t+1} \nabla \boldsymbol{H}_{t+1} = \nabla \boldsymbol{H}_t^{\mathrm{loc}} + \sum_{s=t+1}^{T} \eta_{s,t} \, \nabla \boldsymbol{H}_s^{\mathrm{loc}} + \eta_{T,t} \, \nabla \boldsymbol{H}_T^{\mathrm{ext}}, \tag{23}$$

where $\nabla \boldsymbol{H}_T^{\mathrm{ext}}$ is the upstream gradient of the emitted final state $\boldsymbol{H}_T$ (zero if that state is unused).

The key and gate gradients are linear in this total $\nabla \boldsymbol{H}_t$:

$$\nabla \boldsymbol{k}_t = \boldsymbol{k}_t (\nabla \boldsymbol{H}_t + \nabla \boldsymbol{H}_t^\top), \tag{24}$$

$$\gamma_t \nabla \gamma_t = \nabla log(\gamma_t) = \langle \nabla \boldsymbol{H}_t, \boldsymbol{H}_t - \boldsymbol{k}_t^\top \boldsymbol{k}_t \rangle_F = \langle \nabla \boldsymbol{H}_t, \boldsymbol{H}_t \rangle_F - \tfrac{1}{2} \, (\boldsymbol{k}_t \cdot \nabla \boldsymbol{k}_t), \tag{25}$$

where $\boldsymbol{k}_t \cdot \nabla \boldsymbol{k}_t = 2 \langle \nabla \boldsymbol{H}_t, \boldsymbol{k}_t^\top \boldsymbol{k}_t \rangle_F$. Collecting equation 22 and equation 24, equation 25: $(\boldsymbol{u}_t, \boldsymbol{w}_t, \xi_t)$ define $\nabla \boldsymbol{H}_t^{\mathrm{loc}}$, $\{\nabla \boldsymbol{H}_t\}$ updates gradient through time. $\{\nabla \boldsymbol{H}_t\}$ are then the only quantities needed for $\nabla \boldsymbol{k}_t$ and $\nabla \gamma_t$.

Backward Chunk-wise Parallel. Inside a chunk, $\nabla \boldsymbol{Q}_{[i]}, \boldsymbol{W}_{[i]}, \boldsymbol{U}_{[i]}, \boldsymbol{\xi}_{[i]}$ are determined by chunk-local tensors only: the incoming state $\boldsymbol{H}_{[i]}$, the saved Frobenius scale $\boldsymbol{f}_{[i]}$, and the c tokens of $\boldsymbol{Q}_{[i]}, \boldsymbol{K}_{[i]}, \boldsymbol{O}_{[i]}^{(1)}, \boldsymbol{O}_{[i]}^{(2)}, \nabla \boldsymbol{Q}_{[i]}^*, \boldsymbol{\gamma}_{[i]}$. As equation 20 and equation 21:

$$\begin{aligned} \nabla \boldsymbol{Q}_{[i]} &= a_0 \nabla \boldsymbol{Q}_{[i]}^* + a_1 \mathcal{T}\big((\mathrm{diag}(\boldsymbol{f}_{[i]})^{-1}) \nabla \boldsymbol{Q}_{[i]}^*\big) + a_2 \mathcal{T}\big(\mathcal{T}((\mathrm{diag}(\boldsymbol{f}_{[i]})^{-2}) \nabla \boldsymbol{Q}_{[i]}^*\big), \\ \boldsymbol{W}_{[i]} &= a_2 \mathrm{diag}(\boldsymbol{f}_{[i]}^{-1}) \nabla \boldsymbol{Q}_{[i]}^*, \\ \boldsymbol{U}_{[i]} &= a_1 \mathrm{diag}(\boldsymbol{f}_{[i]}^{-1}) \nabla \boldsymbol{Q}_{[i]}^* + a_2 \mathrm{diag}(\boldsymbol{f}_{[i]}^{-2}) \mathcal{T}(\nabla \boldsymbol{Q}_{[i]}^*), \\ \boldsymbol{\xi}_{[i]} &= -\big(\mathbf{1}_d (a_1 (\boldsymbol{O}_{[i]}^{(1)} \odot \nabla \boldsymbol{Q}_{[i]}^*) + 2a_2 (\boldsymbol{O}_{[i]}^{(2)} \odot \nabla \boldsymbol{Q}_{[i]}^*))^\top\big) \mathrm{diag}(\boldsymbol{f}_{[i]}^{-2}). \end{aligned} \tag{26}$$

Given $\boldsymbol{W}_{[i]}, \boldsymbol{U}_{[i]}, \boldsymbol{\xi}_{[i]}$, the intra-chunk part of equation 24 is four masked Gram products plus the $\xi \boldsymbol{H}$ contractions:

$$\begin{aligned} \nabla \boldsymbol{K}_{[i]} = &\big[\boldsymbol{A}_{[i]} \odot (\boldsymbol{U}_{[i]} \boldsymbol{K}_{[i]}^\top)\big]^\top \boldsymbol{Q}_{[i]} + \big[\boldsymbol{A}_{[i]} \odot (\boldsymbol{Q}_{[i]} \boldsymbol{K}_{[i]}^\top)\big]^\top \boldsymbol{U}_{[i]} \\ &+ \big[\boldsymbol{A}_{[i]} \odot (\boldsymbol{W}_{[i]} \boldsymbol{K}_{[i]}^\top)\big]^\top \boldsymbol{O}_{[i]}^{(1)} + \big[\boldsymbol{A}_{[i]} \odot (\boldsymbol{O}_{[i]}^{(1)} \boldsymbol{K}_{[i]}^\top)\big]^\top \boldsymbol{W}_{[i]} \\ &+ 2 \, (\boldsymbol{\zeta}_{[i]}^\top \mathbf{1}_d) \odot (\boldsymbol{K}_{[i]} \boldsymbol{H}_{[i]}) + 2\big[(\boldsymbol{K}_{[i]} \boldsymbol{K}_{[i]}^\top) \odot \boldsymbol{P}_{[i]}\big]^\top \boldsymbol{K}_{[i]}, \end{aligned} \tag{27}$$

$$\boldsymbol{\zeta}_{[i]} = (\boldsymbol{\xi}_{[i]} \odot \boldsymbol{\eta}_{[i]}) \boldsymbol{A}_{[i]}, \qquad \boldsymbol{P}_{[i]} = \boldsymbol{A}_{[i]}^\top \mathrm{diag}(\boldsymbol{\xi}_{[i]}) \, \boldsymbol{A}_{[i]}. \tag{28}$$

The first four lines expand $(\boldsymbol{u}^\top \boldsymbol{q} + \boldsymbol{q}^\top \boldsymbol{u} + \boldsymbol{w}^\top \boldsymbol{o}^{(1)} + (\boldsymbol{o}^{(1)})^\top \boldsymbol{w})$ against intra-chunk rank-one keys; the last two expand $\xi \boldsymbol{H}$ against $\boldsymbol{H}_t = \gamma_t \boldsymbol{H}_0 + \sum_j A_{tj} \boldsymbol{k}_j^\top \boldsymbol{k}_j$. The same tensors give the outgoing local adjoint for the cross-chunk scan.

Regarding the gradient propagation between blocks, $\nabla \boldsymbol{H}_{[i]}$ will be computed across chunks. According to equation 23, we obtain:

$$\begin{aligned} \nabla \boldsymbol{H}_{[i]} = & \boldsymbol{U}_{[i]}^\top \mathrm{diag}(\boldsymbol{\eta}_{[i]}) \boldsymbol{Q}_{[i]} + \boldsymbol{W}_{[i]}^\top \mathrm{diag}(\boldsymbol{\eta}_{[i]}) \boldsymbol{O}_{[i]}^{(1)} \\ & + (\boldsymbol{\xi}_{[i]} \odot \boldsymbol{\eta}_{[i]}^2) \mathbf{1}_c^\top \boldsymbol{H}_{[i]} + \boldsymbol{K}_{[i]}^\top \mathrm{diag}(\boldsymbol{\zeta}_{[i]}) \boldsymbol{K}_{[i]} + \eta_{(i+1)c,ic} \nabla \boldsymbol{H}_{[i+1]} \end{aligned} \tag{29}$$

Then

$$\nabla \boldsymbol{K}_{[i]} \mathrel{+}= \eta_{(i+1)c,ic} (\mathrm{diag}(\boldsymbol{\eta}_{[i]}))^{-1} \boldsymbol{K}_{[i]} (\nabla \boldsymbol{H}_{[i+1]} + \nabla \boldsymbol{H}_{[i+1]}^\top) \tag{30}$$

The GLA readout equation 17 is differentiated independently of this Gram scan: it produces $\partial \tilde{\boldsymbol{q}}$ and the value gradient $\partial \mathcal{L} / \partial \boldsymbol{v}_t$, routes $\nabla \boldsymbol{x}_t = \alpha_t \, \partial \tilde{\boldsymbol{q}}_t$ into equation 26, and adds $(1 - \alpha_t) \, \partial \tilde{\boldsymbol{q}}_t$ onto equation 20. Its $\nabla \boldsymbol{k}$ is accumulated on top of equation 27 and equation 30. Then use equation 25, get the $\nabla \gamma$. We put the compute in Appendix B.

4 EXPERIMENT

We use the standard and most commonly used stacking architecture like Touvron et al. (2023b). The training code is implemented based on FLA Yang & Zhang (2024). We select Transformer, RetNet, GLA, HGRN2, GatedDeltaNet (GDN), GKA as baseline. We do not use MesaNet due to its instability during the training process Peng et al. (2026).

Table 1: **Results on zero-shot Common-Sense Reasoning Tasks.** † indicates the result is cited from Du et al. (2026); Mei et al. (2026), ‡ indicates the result is obtained through open-source weights in https://huggingface.co/fla-hub. They use the same training setting with us. We conduct evaluations using lm-eval-harness Gao et al. (2024).

Model	**Wiki.** ppl↓	**Lamb.** ppl↓	**ARC-e** acc↑	**ARC-c** $\mathrm{acc_n}$↑	**Hella.** $\mathrm{acc_n}$↑	**Lamb.** acc↑	**PIQA** acc↑	**Wino.** acc↑	**Avg.**
1.3B Params 100B Tokens L=24, d=2048									
Transformer++‡	17.60	19.32	54.97	27.82	49.17	40.77	70.29	55.56	49.76
RetNet‡	18.18	21.97	57.41	26.62	48.07	37.67	69.37	53.59	48.78
HGRN2‡	16.95	15.58	58.25	27.99	51.90	42.36	71.27	52.80	50.76
GLA‡	17.60	19.65	54.97	27.73	48.95	40.00	69.75	54.30	49.28
DeltaNet‡	16.72	15.42	**58.54**	26.79	50.24	42.09	70.51	52.88	50.18
GatedDeltaNet	16.30	14.52	57.87	**29.01**	**52.17**	44.36	**72.03**	**55.09**	**51.76**
GKA	**15.60**	**13.78**	56.82	26.71	51.62	**44.91**	71.00	53.99	50.84
SPLA(ours)	15.66	15.02	57.24	27.65	50.83	43.74	71.27	54.85	50.93

4.1 LANGUAGE MODELING

Experiment setting. We test the model on language modeling. We follow the experimental setup of Zhang et al. (2024); Du et al. (2026) at 1.3B scale (the 340M scale results can be seen in appendix D.1), using the Slimpajama Soboleva et al. (2023) datasets with 100B tokens, and set the training sequence length to 2048. For evaluation, we present zero-shot performance. More detailed introduction can be seen in appendix C.

Common-sense reasoning results. From the commonsense reasoning results in Table 1, it can be seen that compared with GLA and GDN, which update states through a single-step loss, SPLA and GKA, which utilize more information for updating, achieve better results in language modeling loss.

Real world recall task results. In terms of retrieval tasks in Table 2, it can be seen that compared with the other models, GKA and SPLA perform slightly better due to their utilization of more global information.

Table 2: **Results on Recall-Intensive Tasks.** We conduct evaluations using based-lm-eval-harness Arora et al. (2024).

Model	FDA	SWDE	SQUAD	NQ	TriviaQA	Drop	Avg.
1.3B Params 100B Tokens L=24, d=2048							
Transformer++‡	54.68	43.96	43.16	25.34	58.12	21.03	41.04
RetNet‡	20.07	26.99	33.46	16.41	53.14	19.79	28.31
HGRN2‡	13.62	22.68	32.89	19.54	55.51	19.26	27.25
GLA‡	27.16	30.08	34.90	22.20	55.75	19.36	31.58
DeltaNet‡	**42.51**	36.36	34.33	24.55	56.87	21.03	35.94
GatedDeltaNet	40.87	36.73	36.07	25.50	57.94	21.18	36.38
GKA	41.60	**38.52**	**38.86**	25.47	57.35	21.27	**37.17**
SPLA(ours)	40.51	32.90	38.63	**26.77**	**59.89**	**22.57**	36.88

4.2 Ablation

We ablate the performance of SPLA using fixed Taylor expansion, random learnable initialization, and higher orders. The experimental results are shown in Table 3. All results are based on the 340M model. It can be found that allowing the model to perform self-parameter learning leads to better training loss and performance, and allowing higher-order computation also yields better performance, since it can better transform the singular values of $\boldsymbol{K}$.

Table 3: Ablation of different SPLA methods. We compared using fixed Taylor coefficients, learnable parameters, and allowing four term expansion.

Model	Training loss.↓	Wiki.↓	Lamb.↓	CSR-avg	Recall-avg
SPLA-taylor	2.518	25.37	60.21	40.89	24.45
+Learnable parameter	2.513	25.33	61.87	41.41	25.73
+Four term expansion	**2.512**	**25.08**	**53.71**	**41.80**	**26.32**

4.3 Speed test and computation cost

We compare the inference computation cost for GLA, GDN, GKA and ours SPLA. We consider the computational complexity of a single head in a single layer. When calculating FLOPs, we ignore the computational cost of non-d^2 terms and treat both additions and multiplications in the computation as a single operation. In this way, a matrix-vector multiplication costs $2d^2$ FLOPs. In the entire computation flow, after all models obtain the input x of the current layer, they need to use $8d^2$ FLOPs for the corresponding QKVO matrices. The difference lies in the state update. Specifically, GLA computation only involves gated decay of the state and outer-product accumulation. GDN requires an additional retrieval of the state beyond that, and its computation is far less than that of GKA and MesaNet. The computation flow of GKA includes two GLA-like state updates and additionally requires 30 extra Chebyshev iterations for inversion. In comparison, although our method involves a non-local state update, its actual computation is far less than that of GKA. The specific FLOPs and the actual inference time per layer are as Table 4. It can be noted that when comparing GDN and GLA, the methods that require retaining both states have a significant speed change. However, when comparing GKA and MesaNet, due to the reduction in the number of loops, the inference time is about 83% of the original.

Table 4: Comparison of inference FLOPs and speed time using one layer with BHD=[1,8,128].

Model	Computation Flops	Inference Time Cost(ms)
GLA	$13d^2$	0.794
GDN	$16d^2$	0.806
GKA	$76d^2$	1.168
SPLA	$22d^2$	0.979

4.4 VISIBLE OF SINGULAR VALUE

We compare the matrix singular value transformation functions obtained by SPLA after training with those of the Muon and GKA methods. Since the SPLA blocks in different layers and heads have different transformations, we select some representative ones and plot them in Figure 2. We also visualize the condition numbers of $\boldsymbol{K}'$ for different models as a function of position in the Figure 3.

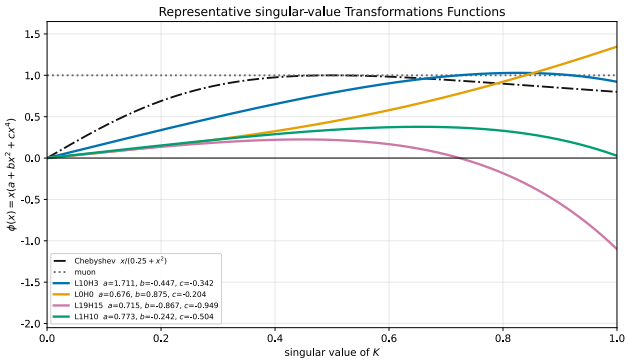

Figure 2: Representative Singular Value Transformation Function

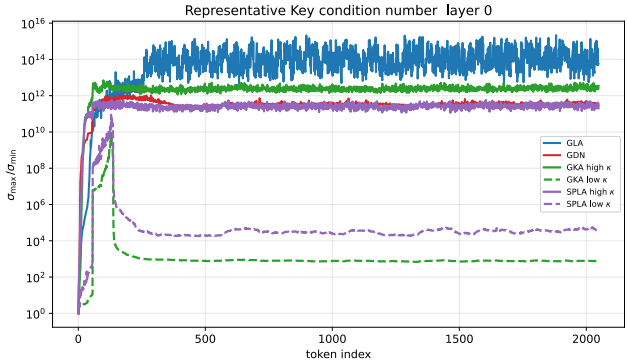

Figure 3: Visible of condition numbers of $\boldsymbol{K}'$. For GLA, we progressively apply decay to each dimension. For GDN, we transform $\boldsymbol{K}$ using its positional encoding scheme, which is equivalent to that of Path Yang et al. (2025b). For GKA and SPLA, considering the symmetry of the Gram matrix, we move the operator originally applied to the query onto $\boldsymbol{K}$. Since SPLA and GKA exhibit notably different behaviors across different heads, we plot two curves for each of them.

As shown in the Figure 2, by autonomously learning the coefficients of different orders, the model enables the singular value transformation functions to exhibit different behaviors. Figure 3 shows the constraint of GKA and SPLA on the matrix condition number at certain heads, effectively suppressing the dominant position of the main direction.

5 CONCLUSION

In this work, we analyze some recent linear model studies and find that they essentially perform singular value spectrum transformations on all key-value matrices. We propose a method that uses polynomial expansion to perform singular value transformation, enabling the model to more selectively control the behavior of the current state matrix, which leads to SPLA. Similar to MesaNet and GKA, SPLA can utilize longer-range past information for state updates, thereby achieving strong capabilities compared with methods that use single-step state updates. In contrast to methods that use CG iterations and Chebyshev iterations for full inversion, our proposed method brings lower computational overhead in both training and inference, effectively reducing computation.

AI USE STATEMENT

In this work, we used generative AI tools for language polishing of the paper to improve its readability. In terms of experiments, we used AI to help write the core Triton operators in the code to improve the training speed of the model. We did not use generative AI tools to search for research ideas, design or conduct experiments, or provide related interpretations of the work. All AI-assisted content has been reviewed by authors, we take responsibility for the final content of this work.

ETHICS STATEMENT

This paper proposes a fundamental module as the underlying architecture of large language models. We do not involve human subjects, data collection or publication. We believe that this study does not have any clear ethical issues that need to be pointed out

REPRODUCIBILITY STATEMENT

We have elaborated on the ideas and experimental procedures of the paper in detail, and we place the corresponding experimental setups in the main text 4 and the appendix C. We will provide the corresponding anonymous code, and upon final acceptance of the paper, we will provide the complete code repository link and release the training weights.

REFERENCES

Simran Arora, Brandon Yang, Sabri Eyuboglu, Avanika Narayan, Andrew Hojel, Immanuel Trummer, and Christopher Ré. Language models enable simple systems for generating structured views of heterogeneous data lakes. *arXiv preprint arXiv:2304.09433*, 2023.

Simran Arora, Aman Timalsina, Aaryan Singhal, Sabri Eyuboglu, Xinyi Zhao, Ashish Rao, Atri Rudra, and Christopher Re. Just read twice: closing the recall gap for recurrent language models. In *Workshop on Efficient Systems for Foundation Models II@ ICML2024*, 2024.

Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xlstm: Extended long short-term memory. *Advances in Neural Information Processing Systems*, 37:107547–107603, 2024.

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 *Forty-third International Conference on Machine Learning*, 2026.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

Yingfa Chen, Zhen Leng Thai, Zihan Zhou, Zhu Zhang, Xingyu Shen, Shuo Wang, Chaojun Xiao, Xu Han, and Zhiyuan Liu. Hybrid linear attention done right: Efficient distillation and effective architectures for extremely long contexts. *arXiv preprint arXiv:2601.22156*, 2026.

Yuhong Chou, Man Yao, Kexin Wang, Yuqi Pan, Rui-Jie Zhu, Jibin Wu, Yiran Zhong, Yu Qiao, Bo Xu, and Guoqi Li. Metala: Unified optimal linear approximation to softmax attention map. *Advances in Neural Information Processing Systems*, 37:71034–71067, 2024.

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 arXiv:1803.05457*, 2018.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, G Heigold, S Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of 9th International Conference on Learning Representations*, 2020.

Jusen Du, Weigao Sun, Disen Lan, Jiaxi Hu, Zhang Tao, and Yu Cheng. Mom: Linear sequence modeling with mixture-of-memories. In *International Conference on Learning Representations*, volume 2026, pp. 106613–106631, 2026.

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

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, 2024.

Riccardo Grazzi, Julien Siems, Arber Zela, Jörg KH Franke, Frank Hutter, and Massimiliano Pontil. Unlocking state-tracking in linear rnns through negative eigenvalues. In *The Thirteenth International Conference on Learning Representations*, 2025.

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

Albert Gu, Karan Goel, Ankit Gupta, and Christopher Ré. On the parameterization and initialization of diagonal state space models. *Advances in Neural Information Processing Systems*, 35:35971–35983, 2022a.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *Proceedings of 10th International Conference on Learning Representations*, 2022b.

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 *International Conference on Machine Learning*, pp. 9639–9659. PMLR, 2022.

Samy Jelassi, David Brandfonbrener, Sham M Kakade, and Eran Malach. Repeat after me: transformers are better than state space models at copying. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 21502–21521, 2024.

Keller Jordan, Yuchen Jin, Vlado Boza, You Jiacheng, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL `https://kellerjordan.github.io/posts/muon/`.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, 2017.

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

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

Bo Liu, Rui Wang, Lemeng Wu, Yihao Feng, Peter Stone, and Qiang Liu. Longhorn: State space models are amortized online learners. *arXiv preprint arXiv:2407.14207*, 2024.

Colin Lockard, Prashant Shiralkar, and Xin Luna Dong. Openceres: When open information extraction meets the semi-structured web. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 3047–3056, 2019.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Shijie Mei, Man Yao, Jiabo Tong, Bo XU, and Guoqi Li. Head-in-head in linear attention. In *Forty-third International Conference on Machine Learning*, 2026.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *Proceedings of 6th International Conference on Learning Representations*, 2017.

William Merrill, Jackson Petty, and Ashish Sabharwal. The illusion of state in state-space models. 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. 35492–35506. PMLR, 21–27 Jul 2024.

Thai-Khanh Nguyen, Ngoc-Bich-Uyen Vo, Thieu N Vo, Tan M Nguyen, and Cuong Pham. Muon-ssm: Orthogonalizing state space models for sequence modeling. In *Forty-third International Conference on Machine Learning*, 2026.

Yuqi Pan, Yongqi An, Zheng Li, Yuhong Chou, Rui-Jie Zhu, Xiaohui Wang, Mingxuan Wang, Jinqiao Wang, and Guoqi Li. Scaling linear attention capacity with sparse state expansion. In *International Conference on Learning Representations*, volume 2026, pp. 130348–130376, 2026.

D Paperno, G Kruszewski, A Lazaridou, QN Pham, Raffaella Bernardi, S Pezzelle, M Baroni, G Boleda, and R Fernández. The lambada dataset: Word prediction requiring a broad discourse context. In *54th Annual Meeting of the Association for Computational Linguistics, ACL 2016-Long Papers*, volume 3, pp. 1525–1534, 2016.

Liangzu Peng, Aditya Chattopadhyay, Luca Zancato, Elvis Nunez, Wei Xia, and Stefano Soatto. Gated kalmanet: A fading memory layer through test-time ridge regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20212–20222, 2026.

Zhen Qin, Songlin Yang, and Yiran Zhong. Hierarchically gated recurrent neural network for sequence modeling. *Advances in Neural Information Processing Systems*, 36:33202–33221, 2023.

Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. Hgrn2: Gated linear rnns with state expansion. In *First Conference on Language Modeling*, 2024.

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

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

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

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

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International conference on machine learning*, pp. 9355–9366. PMLR, 2021.

Vaisakh Shaj, Cameron Barker, Aidan Scannell, Andras Szecsenyi, Elliot J. Crowley, and Amos Storkey. Kalman linear attention: Parallel bayesian filtering for efficient language modeling and state tracking. In *Forty-third International Conference on Machine Learning*, 2026.

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

Jianlin Su. Violent aesthetics in matrix function approximation, Jun 2026. URL https://spaces.ac.cn/archives/11787.

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 *Proceedings of the 42nd International Conference on Machine Learning*, ICML'25. JMLR.org, 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, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

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

Johannes Von Oswald, Maximilian Schlegel, Alexander Meulemans, Seijin Kobayashi, Eyvind Niklasson, Nicolas Zucchet, Nino Scherrer, Nolan Miller, Mark Sandler, Max Vladymyrov, et al. Uncovering mesa-optimization algorithms in transformers. *arXiv preprint arXiv:2309.05858*, 2023.

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.

Ke Alexander Wang, Jiaxin Shi, and Emily B Fox. Test-time regression: a unifying framework for designing sequence models with associative memory. *arXiv preprint arXiv:2501.12352*, 2025.

Songlin Yang and Yu Zhang. Fla: A triton-based library for hardware-efficient implementations of linear attention mechanism, 2024. URL https://github.com/fla-org/flash-linear-attention.

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

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. *Advances in Neural Information Processing Systems*, 37:115491–115522, 2024b.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *The Thirteenth International Conference on Learning Representations*, 2025a.

Songlin Yang, Yikang Shen, Kaiyue Wen, Shawn Tan, Mayank Mishra, Liliang Ren, Rameswar Panda, and Yoon Kim. Path attention: Position encoding via accumulating householder transformations. *arXiv preprint arXiv:2505.16381*, 2025b.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, 2019.

Yu Zhang, Songlin Yang, Rui-Jie Zhu, Yue Zhang, Leyang Cui, Yiqiao Wang, Bolun Wang, Freda Shi, Bailin Wang, Wei Bi, et al. Gated slot attention for efficient linear-time sequence modeling. *Advances in Neural Information Processing Systems*, 37:116870–116898, 2024.

A FORWARD CHUNK-WISE PARALLEL WITH FROBENIUS NORM

The Gram recurrence is

$$\boldsymbol{H}_t = \gamma_t \boldsymbol{H}_{t-1} + \boldsymbol{k}_t^\top \boldsymbol{k}_t, \qquad \boldsymbol{H}_0 = \boldsymbol{0}. \tag{31}$$

With $f_t = \|\boldsymbol{H}_t\|_F + \varepsilon$ and $\widetilde{\boldsymbol{H}}_t = \boldsymbol{H}_t / f_t$, the three-term map is

$$\begin{aligned} \boldsymbol{o}_t^{(1)} &= \boldsymbol{q}_t \boldsymbol{H}_t / f_t, \\ \boldsymbol{o}_t^{(2)} &= \boldsymbol{o}_t^{(1)} \boldsymbol{H}_t / f_t = \boldsymbol{q}_t \boldsymbol{H}_t^2 / f_t^2, \\ \boldsymbol{q}_t^* &= a_0\, \boldsymbol{q}_t + a_1\, \boldsymbol{o}_t^{(1)} + a_2\, \boldsymbol{o}_t^{(2)}. \end{aligned} \tag{32}$$

The mixed readout is unchanged:

$$\tilde{\boldsymbol{q}}_t = (1-\alpha_t)\, \boldsymbol{q}_t + \alpha_t\, \boldsymbol{q}_t^*, \qquad \boldsymbol{y}_t = \mathrm{GLA}(\tilde{\boldsymbol{q}}_t, \boldsymbol{k}_t, \boldsymbol{v}_t, g_t^{(kv)}). \tag{33}$$

Unrolling equation 31 inside chunk i gives

$$\boldsymbol{H}_t = \eta_{t,ic}\, \boldsymbol{H}_{[i]} + \sum_{j=ic+1}^{t} \eta_{t,j}\, \boldsymbol{k}_j^\top \boldsymbol{k}_j. \tag{34}$$

The same identity applied to a $c \times d$ stream $\boldsymbol{Z}_{[i]}$ is

$$\mathcal{T}(\boldsymbol{Z}_{[i]}) = (\boldsymbol{\eta}_{[i]}^\top \boldsymbol{1}_d) \odot (\boldsymbol{Z}_{[i]} \boldsymbol{H}_{[i]}) + \big(\boldsymbol{A}_{[i]} \odot (\boldsymbol{Z}_{[i]} \boldsymbol{K}_{[i]}^\top)\big) \boldsymbol{K}_{[i]}. \tag{35}$$

The squared Frobenius norms of the c token states are the row vector

$$\begin{aligned} \boldsymbol{z}_{[i]} &= \boldsymbol{1}_d \big((\boldsymbol{K}_{[i]} \boldsymbol{H}_{[i]}) \odot \boldsymbol{K}_{[i]}\big)^\top, \qquad \boldsymbol{S}_{kk[i]} = \boldsymbol{K}_{[i]} \boldsymbol{K}_{[i]}^\top, \\ \boldsymbol{n}_{[i]} &= (\boldsymbol{\eta}_{[i]} \odot \boldsymbol{\eta}_{[i]})\, \|\boldsymbol{H}_{[i]}\|_F^2 + 2\, \boldsymbol{\eta}_{[i]} \odot (\boldsymbol{z}_{[i]} \boldsymbol{A}_{[i]}^\top) + \text{row-sum}\big((\boldsymbol{A}_{[i]} (\boldsymbol{S}_{kk[i]} \odot \boldsymbol{S}_{kk[i]})) \odot \boldsymbol{A}_{[i]}\big), \\ \boldsymbol{f}_{[i]} &= \sqrt{\boldsymbol{n}_{[i]}} + \varepsilon \in \mathbb{R}^{1 \times c}. \end{aligned} \tag{36}$$

B BACKWARD CHUNK-WISE PARALLEL

Local statistics. Inside chunk i,

$$\begin{aligned} \nabla \boldsymbol{Q}_{[i]} &= a_0 \nabla \boldsymbol{Q}_{[i]}^* + a_1 \mathrm{diag}(\boldsymbol{f}_{[i]})^{-1}\, \mathcal{T}(\nabla \boldsymbol{Q}_{[i]}^*) + a_2 \mathrm{diag}(\boldsymbol{f}_{[i]})^{-1}\, \mathcal{T}\big(\mathrm{diag}(\boldsymbol{f}_{[i]})^{-1}\, \mathcal{T}(\nabla \boldsymbol{Q}_{[i]}^*)\big), \\ \boldsymbol{W}_{[i]} &= a_2 \mathrm{diag}(\boldsymbol{f}_{[i]})^{-1} \nabla \boldsymbol{Q}_{[i]}^*, \\ \boldsymbol{U}_{[i]} &= a_1 \mathrm{diag}(\boldsymbol{f}_{[i]})^{-1} \nabla \boldsymbol{Q}_{[i]}^* + a_2 \mathrm{diag}(\boldsymbol{f}_{[i]})^{-2}\, \mathcal{T}(\nabla \boldsymbol{Q}_{[i]}^*), \\ \boldsymbol{\xi}_{[i]} &= -\big(\boldsymbol{1}_d \big(a_1 (\boldsymbol{O}_{[i]}^{(1)} \odot \nabla \boldsymbol{Q}_{[i]}^*) + 2a_2 (\boldsymbol{O}_{[i]}^{(2)} \odot \nabla \boldsymbol{Q}_{[i]}^*)\big)^\top\big) \mathrm{diag}(\boldsymbol{f}_{[i]})^{-2}. \end{aligned} \tag{37}$$

Intra-chunk $\nabla \boldsymbol{K}$ and $\langle \nabla \boldsymbol{H}, \boldsymbol{H} \rangle$. These terms come from $\nabla \boldsymbol{H}^{\mathrm{loc}}$ only. Define

$$\boldsymbol{\zeta}_{[i]} = (\boldsymbol{\xi}_{[i]} \odot \boldsymbol{\eta}_{[i]}) \boldsymbol{A}_{[i]}, \qquad \boldsymbol{P}_{[i]} = \boldsymbol{A}_{[i]}^\top \mathrm{diag}(\boldsymbol{\xi}_{[i]})\, \boldsymbol{A}_{[i]}. \tag{38}$$

Then

$$\begin{aligned} \nabla \boldsymbol{K}_{[i]} = {} & \big[\boldsymbol{A}_{[i]} \odot (\boldsymbol{U}_{[i]} \boldsymbol{K}_{[i]}^\top)\big]^\top \boldsymbol{Q}_{[i]} + \big[\boldsymbol{A}_{[i]} \odot (\boldsymbol{Q}_{[i]} \boldsymbol{K}_{[i]}^\top)\big]^\top \boldsymbol{U}_{[i]} \\ & + \big[\boldsymbol{A}_{[i]} \odot (\boldsymbol{W}_{[i]} \boldsymbol{K}_{[i]}^\top)\big]^\top \boldsymbol{O}_{[i]}^{(1)} + \big[\boldsymbol{A}_{[i]} \odot (\boldsymbol{O}_{[i]}^{(1)} \boldsymbol{K}_{[i]}^\top)\big]^\top \boldsymbol{W}_{[i]} \\ & + 2\, (\boldsymbol{\zeta}_{[i]}^\top \boldsymbol{1}_d) \odot (\boldsymbol{K}_{[i]} \boldsymbol{H}_{[i]}) + 2\big[(\boldsymbol{K}_{[i]} \boldsymbol{K}_{[i]}^\top) \odot \boldsymbol{P}_{[i]}\big]^\top \boldsymbol{K}_{[i]}. \end{aligned} \tag{39}$$

Due to:

$$\nabla \boldsymbol{H}_t^{\mathrm{loc}} = \boldsymbol{u}_t^\top \boldsymbol{q}_t + \boldsymbol{w}_t^\top \boldsymbol{o}_t^{(1)} + \xi_t\, \boldsymbol{H}_t \tag{40}$$

$$\gamma_t \nabla \gamma_t = \nabla log(\gamma_t) = \langle \nabla \boldsymbol{H}_t, \boldsymbol{H}_t - \boldsymbol{k}_t^\top \boldsymbol{k}_t \rangle_F = \langle \nabla \boldsymbol{H}_t, \boldsymbol{H}_t \rangle_F - \tfrac{1}{2}\, (\boldsymbol{k}_t \cdot \nabla \boldsymbol{k}_t), \tag{41}$$

Because $\boldsymbol{H}_t$ is symmetric, $\langle \nabla \boldsymbol{H}_t^{loc}, \boldsymbol{H}_t \rangle_F$ can be calculated by:

$$p_t = \langle \nabla \boldsymbol{H}_t^{\mathrm{loc}}, \boldsymbol{H}_t \rangle_F = (\boldsymbol{q}_t \boldsymbol{H}_t) \cdot \boldsymbol{u}_t + (\boldsymbol{o}_t^{(1)} \boldsymbol{H}_t) \cdot \boldsymbol{w}_t + \xi_t \|\boldsymbol{H}_t\|_F^2. \tag{42}$$

With $\boldsymbol{o}_t^{(1)} = \boldsymbol{q}_t \boldsymbol{H}_t / f_t$ and $\boldsymbol{o}_t^{(2)} = \boldsymbol{o}_t^{(1)} \boldsymbol{H}_t / f_t$, the first two dots are $f_t(\boldsymbol{o}_t^{(1)} \cdot \boldsymbol{u}_t)$ and $f_t(\boldsymbol{o}_t^{(2)} \cdot \boldsymbol{w}_t)$. The squared norms $\boldsymbol{n}_{[i]}$ are those already formed in equation 36, before adding ε, so

$$\boldsymbol{p}_{[i]} = \boldsymbol{f}_{[i]} \odot \Big(\mathbf{1}_d \big((\boldsymbol{O}_{[i]}^{(1)} \odot \boldsymbol{U}_{[i]} + \boldsymbol{O}_{[i]}^{(2)} \odot \boldsymbol{W}_{[i]})^\top \big) \Big) + \boldsymbol{\xi}_{[i]} \odot \boldsymbol{n}_{[i]}. \tag{43}$$

The same row is obtained by splitting $\boldsymbol{H}_t$ as in equation 34. Only row t of $\boldsymbol{A}_{[i]}$ enters the inner product at time t; $\boldsymbol{\zeta}_{[i]} = (\boldsymbol{\xi}_{[i]} \odot \boldsymbol{\eta}_{[i]}) \boldsymbol{A}_{[i]}$ sums over time and is already consumed by $\nabla \boldsymbol{K}$ and $\nabla \boldsymbol{H}$.

$$\begin{aligned} \boldsymbol{\rho}_{[i]} &= \mathbf{1}_d \big((\boldsymbol{Q}_{[i]} \boldsymbol{H}_{[i]}) \odot \boldsymbol{U}_{[i]} + (\boldsymbol{O}_{[i]}^{(1)} \boldsymbol{H}_{[i]}) \odot \boldsymbol{W}_{[i]} \big)^\top, \\ \boldsymbol{M}_{[i]} &= (\boldsymbol{K}_{[i]} \boldsymbol{U}_{[i]}^\top) \odot (\boldsymbol{K}_{[i]} \boldsymbol{Q}_{[i]}^\top) + (\boldsymbol{K}_{[i]} \boldsymbol{W}_{[i]}^\top) \odot \big(\boldsymbol{K}_{[i]} (\boldsymbol{O}_{[i]}^{(1)})^\top \big), \\ \boldsymbol{p}_{[i]} &= \boldsymbol{\eta}_{[i]} \odot \boldsymbol{\rho}_{[i]} + \text{row-sum}(\boldsymbol{A}_{[i]} \odot \boldsymbol{M}_{[i]}^\top) + \boldsymbol{\xi}_{[i]} \odot \boldsymbol{n}_{[i]}. \end{aligned} \tag{44}$$

Here $\boldsymbol{\eta}_{[i]} \odot \boldsymbol{\rho}_{[i]}$ is the incoming state, the row sum is $\sum_j A_{tj} \langle \boldsymbol{u}_t^\top \boldsymbol{q}_t + \boldsymbol{w}_t^\top \boldsymbol{o}_t^{(1)}, \boldsymbol{k}_j^\top \boldsymbol{k}_j \rangle_F$, and $\boldsymbol{\xi}_{[i]} \odot \boldsymbol{n}_{[i]}$ is the full $\xi_t \|\boldsymbol{H}_t\|_F^2$.

This $\boldsymbol{p}_{[i]}$ is not enough yet. We also need to calculate the gradient $\nabla \boldsymbol{H}_t$ obtained through transmission.

Cross-chunk $\nabla \boldsymbol{H}$. The local adjoint of the incoming state $\boldsymbol{H}_{[i]}$ is

$$\begin{aligned} \nabla \boldsymbol{H}_{[i]}^{local} = \boldsymbol{U}_{[i]}^\top \mathrm{diag}(\boldsymbol{\eta}_{[i]}) \boldsymbol{Q}_{[i]} + \boldsymbol{W}_{[i]}^\top \mathrm{diag}(\boldsymbol{\eta}_{[i]}) \boldsymbol{O}_{[i]}^{(1)} \\ + (\boldsymbol{\xi}_{[i]} \odot \boldsymbol{\eta}_{[i]}^2) \mathbf{1}_c^\top \boldsymbol{H}_{[i]} + \boldsymbol{K}_{[i]}^\top \mathrm{diag}(\boldsymbol{\zeta}_{[i]}) \boldsymbol{K}_{[i]}. \end{aligned} \tag{45}$$

The reverse scan uses the outgoing adjoint $\nabla \boldsymbol{H}_{[i+1]} = \nabla \boldsymbol{H}_{(i+1)c}$,

$$\nabla \boldsymbol{H}_{[i]} = \nabla \boldsymbol{H}_{[i]}^{local} + \eta_{(i+1)c,ic} \nabla \boldsymbol{H}_{[i+1]}, \tag{46}$$

evaluated from the last chunk to the first, with $\nabla \boldsymbol{H}_{[N]} = \nabla \boldsymbol{H}_T^{\mathrm{ext}}$. Equation equation 46 is the adjoint of the incoming $\boldsymbol{H}_{[i]}$; the outgoing $\nabla \boldsymbol{H}_{[i+1]}$ is injected into chunk i as follows.

Boundary $\nabla \boldsymbol{K}$ and $\nabla \boldsymbol{\gamma}$. Let $\boldsymbol{\eta}'_{[i]}$ be the last row of $\boldsymbol{A}_{[i]}$, equivalently $\boldsymbol{\eta}'_{[i]} = \eta_{(i+1)c,ic} (\boldsymbol{\eta}_{[i]})^{-1}$.

$$\nabla \boldsymbol{K}_{[i]} \mathrel{+}= (\mathrm{diag}(\boldsymbol{\eta}'_{[i]})) \boldsymbol{K}_{[i]} \big(\nabla \boldsymbol{H}_{[i+1]} + \nabla \boldsymbol{H}_{[i+1]}^\top \big). \tag{47}$$

The same outgoing matrix contracts against $\boldsymbol{H}_t$ by equation 34,

$$\begin{aligned} \sigma_{[i]} &= \langle \nabla \boldsymbol{H}_{[i+1]}, \boldsymbol{H}_{[i]} \rangle_F, \\ \boldsymbol{v}_{[i]} &= \mathbf{1}_d \big((\boldsymbol{K}_{[i]} \nabla \boldsymbol{H}_{[i+1]}) \odot \boldsymbol{K}_{[i]} \big)^\top, \\ \boldsymbol{h}_{[i]} &= \sigma_{[i]} \boldsymbol{\eta}_{[i]} + \boldsymbol{v}_{[i]} \boldsymbol{A}_{[i]}^\top. \end{aligned} \tag{48}$$

After get the total $\nabla \boldsymbol{k}$,we can get $\nabla \gamma$ by:

$$\boldsymbol{\gamma}_{[i]} \odot \nabla \boldsymbol{\gamma}_{[i]} = \boldsymbol{p}_{[i]} + \boldsymbol{\eta}'_{[i]} \odot \boldsymbol{h}_{[i]} - \tfrac{1}{2} \mathbf{1}_d (\boldsymbol{K}_{[i]} \odot \nabla \boldsymbol{K}_{[i]})^\top. \tag{49}$$

The last term uses the total $\nabla \boldsymbol{K}_{[i]}$ (intra plus boundary), so it converts both $\boldsymbol{p}_{[i]}$ and $\boldsymbol{\eta}'_{[i]} \odot \boldsymbol{h}_{[i]}$ from $\langle \nabla \boldsymbol{H}, \boldsymbol{H}_t \rangle_F$ into $\langle \nabla \boldsymbol{H}, \boldsymbol{H}_t - \boldsymbol{k}^\top \boldsymbol{k} \rangle_F$.

C EXPERIMENT SETTING

For all models, we use SlimPajama datasets Soboleva et al. (2023) with Llama2 tokenizer with a vocabulary size of 32000. We set training context length with 2048, learning rate as 3e-4 with a warm-up cosine annealing scheduler, the minimum learning rate is 3e-5, 0.1 of the peak learning rate, the optimizer is AdamW Loshchilov & Hutter (2017) with $\beta s = (0.9, 0.95)$, weight decay 0.01,max grad norm 1.0, the other different training setting can be seen in Table 5. For SPLA and GKA, we set their head dimension to 128 and the rest of the models to 256, because their internal shared memory settings cannot continue to expand the head dimension size.

Regarding evaluation, all platforms used in our experiments have been listed in the main text. For generation tasks, we limit the maximum input length to 2000 tokens and allow up to 48 tokens in the output.

Table 5: Training Setting of Pretrain Models.

Parameters	340M	1.3B
Layers	24	24
Dimension	1024	2048
Tokens	15B	100B
Total batch size	256	1024
Total steps	30720	50016
Warm up steps	512	512
Tied Word Embedding	True	False

Datasets Description. The commonsense reasoning tasks consist of WikiText Merity et al. (2017), Lambada Standard Paperno et al. (2016), ARC-Easy and ARC-challenge Clark et al. (2018), HellaSwag Zellers et al. (2019), PIQA Bisk et al. (2020), WinoGrande Sakaguchi et al. (2021). Real world recall tasks consist of FDA Arora et al. (2023), SWDE Lockard et al. (2019), SQUAD Rajpurkar et al. (2018), NQ Kwiatkowski et al. (2019), TriviaQA Joshi et al. (2017), Drop Dua et al. (2019).

D EXPERIMENT

D.1 340M SCALE RESULTS.

We put the results of 340M scale training in Table 6 and 7.

Table 6: **340M Results on zero-shot Common-Sense Reasoning Tasks.**

Model	**Wiki.** ppl↓	**Lamb.** ppl↓	**ARC-e** acc↑	**ARC-c** acc_n↑	**Hella.** acc_n↑	**Lamb.** acc↑	**PIQA** acc↑	**Wino.** acc↑	**Avg.**
340M Params 15B Tokens L=24, d=1024									
Transformer++†	26.88	76.46	44.91	25.94	34.95	26.90	64.31	51.07	41.35
RetNet†	31.07	87.11	44.49	23.04	33.86	23.93	63.49	52.33	40.19
HGRN2†	27.90	77.40	45.24	23.63	35.61	24.74	65.45	**54.06**	41.46
GLA†	28.78	79.95	44.53	22.27	34.84	24.94	63.93	51.38	40.32
DeltaNet	27.72	71.04	46.13	**25.68**	34.90	24.32	64.69	51.30	41.17
GatedDeltaNet	26.32	56.03	**46.30**	23.55	**35.78**	27.36	**65.61**	52.17	41.79
GKA	25.19	59.71	**46.30**	23.81	35.21	27.27	64.69	51.22	41.42
SPLA(ours)	**25.08**	**53.71**	45.33	22.87	35.54	**28.95**	65.34	52.80	**41.80**

Table 7: **340M Results on Recall-Intensive Tasks.**

Model	**FDA**	**SWDE**	**SQUAD**	**NQ**	**TriviaQA**	**Drop**	**Avg.**
340M Params 15B Tokens L=24, d=1024							
Transformer++†	46.14	25.87	33.22	18.94	45.97	20.03	31.70
RetNet†	5.90	9.28	22.41	6.91	40.05	18.59	17.19
HGRN2†	11.53	17.34	24.08	12.67	43.84	17.35	21.14
GLA†	11.26	16.78	27.85	12.77	43.90	17.68	21.71
DeltaNet	**30.60**	25.40	25.66	15.30	42.77	19.45	**26.53**
GatedDeltaNet	24.25	23.71	27.88	14.28	**45.91**	18.44	24.88
GKA	19.98	**26.52**	**31.00**	14.82	44.49	**20.80**	26.27
SPLA(ours)	21.89	25.87	28.18	**16.41**	44.79	**20.80**	26.32

D.2 USE MUON OPTIMIZER.

The use of Muon optimizer can significantly improve the performance of the model. Considering fairness, we use the original adamW optimizer in the main text and list the effects of using Muon in Table 8.

Table 8: Ablation of using Muon optimizer. We use 340M model and 15B tokens. Muon training is applied only to all linear layers, with a learning rate of 0.02 and momentum set to 0.95, while the original embedding layer and normalization parts keep the original AdamW and learning rate unchanged.

Model	Training loss.↓	Wiki.↓	Lamb.↓	CSR-avg	Recall-avg
SPLA	2.51265	25.08	53.71	41.80	26.32
+Muon	**2.4271**	**22.12**	**30.84**	**44.10**	**30.70**