Improving Bilinear RNNs with Closed-loop Control

Jiaxi Hu[1] **Yongqi Pan**[1 *] **Jusen Du**[2] **Disen Lan**[2] **Xiaqiang Tang**[1] **Qingsong Wen**[3]
Yuxuan Liang[1 ✉] **Weigao Sun**[2 ✉]

[1]The Hong Kong University of Science and Technology (Guangzhou)
[2]Shanghai AI Laboratory [3]Squirrel Ai Learning, USA

Abstract

Recent efficient sequence modeling methods such as Gated DeltaNet, TTT, and RWKV-7 have achieved performance improvements by supervising the recurrent memory management through Delta learning rule. Unlike previous state-space models (e.g., Mamba) and gated linear attentions (e.g., GLA), these models introduce interactions between the recurrent state and the key vector, structurally resembling bilinear systems. In this paper, we first introduce the concept of Bilinear RNNs with a comprehensive analysis on the advantages and limitations of these models. Then, based on closed-loop control theory, we propose a novel Bilinear RNN variant named Comba, which adopts a scalar-plus-low-rank state transition, with both state feedback and output feedback corrections. We also implement a hardware-efficient chunk-wise parallel kernel in Triton and train models with 340M/1.3B parameters on large-scale corpus. Comba demonstrates superior performance and computation efficiency in both language and vision modeling.

`https://github.com/fla-org/flash-linear-attention`.

"Learning without thinking misleads."

– Confucius, 479 BC

1 Introduction

Autoregressive Transformers [95] have become a foundation of modern AI, primarily due to the efficient parallel computation made possible by softmax-based self-attention. This mechanism enables effective memory scaling by directly appending `key` and `value` vectors into the KV cache, which contributes to strong performance on tasks such as in-context learning and long-context retrieval. However, this design also comes with challenges, including quadratic time complexity and unbounded memory growth during inference [59], which limits the model's scalability for long-sequence tasks. To address these, numerous improvements have been introduced, including sliding window attention [8, 25], sparse attention techniques [64, 111, 103], and efficient KV cache management [59].

Meanwhile, efficient sequence-mixing approaches such as gated linear attention [112, 79, 22, 107, 105, 5] and selective state space models (SSMs) [38, 26, 84] offer a compelling alternative. These models aim to establish a linear *key-value associative memory* [105, 34] register with constant states and data-(in)dependent gating ($\boldsymbol{\alpha}$, $\boldsymbol{\beta}$), as presented by $\boldsymbol{S}_t = \boldsymbol{\alpha}_{(t)}\boldsymbol{S}_{t-1} + \boldsymbol{\beta}_{(t)}\boldsymbol{v}_t\boldsymbol{k}_t^\intercal$. The inherent recurrent structure of these models enables them to maintain constant memory overhead and $\mathcal{O}(1)$ time complexity during inference. Despite these models originating from distinct theoretical frameworks, for example, early linear attentions [53, 82] like Linformer [98] attempt to reformulate quadratic attention computation as $\boldsymbol{O} = \phi(\boldsymbol{Q})(\phi(\boldsymbol{K})^\intercal\boldsymbol{V})$ with kernel mapping ϕ, while original SSMs [39, 42] like S4 [41] intent to parameterize a continuous dynamical system to a discrete form, recent literature [22, 7, 52, 6] have unified these models to the concept of Linear RNNs.

✉Corresponding authors (yuxliang@outlook.com, sunweigao@outlook.com). * Work done during Yongqi Pan's internship at Hong Kong University of Science and Technology (Guangzhou).

Table 1: **Memorizing Mechanismes in state-of-the-art Sequence Modeling Methods.** Softmax Attention ensures precise memory storage, while sliding window attention constrains storage space. Linear RNNs (lines 3-6) use data-(in)dependent gate for unsupervised memory management, with Mamba2 approximating the forget gate α to 1, forcing the model to forget. Bilinear RNNs (lines 7-9) is no longer a linear *key-value memory register* $\boldsymbol{S}_{t+1} = (\boldsymbol{\alpha}, \boldsymbol{\beta})@(\boldsymbol{S}_t, \boldsymbol{k}_t^\intercal \boldsymbol{v}_t)^\intercal$, which supervise the management process based on the Delta learning rule – effectively equivalent to minimizing a Stochastic Gradient Descent $\nabla_S \|\boldsymbol{S}_t\boldsymbol{k}_t - \boldsymbol{v}_t\|^2$. When ignoring layer normalization and residual components, TTT-Linear can also be categorized as a bilinear RNN. As modern Nonlinear RNNs, MIRAS and its variants (e.g., TTT-MLP, Titans [7], Lattice [52]) have stronger expressiveness because of the nonlinearity g, high-order optimizations, and MLP-based deep memory, but are limited by the chunk-wise parallelism (§3). Our proposed Comba further improves Bilinear RNNs by closed-loop control.

Model	**Memorizing with Gate**	**Optimization Objective $\mathcal{L}$**
Softmax Attention: Conductivist-based infinite key-value associative memory registers		
SA [95]	$\boldsymbol{S}_t = \boldsymbol{S}_{t-1}.\,\mathrm{append}(\boldsymbol{k}_t, \boldsymbol{v}_t)$	$\sum_{i=1}^{t} \exp(\boldsymbol{q}_t^\intercal \boldsymbol{k}_i)\,\|\boldsymbol{v} - \boldsymbol{v}_i\|^2$ [97]
SWA [8]	$\boldsymbol{S}_t = \boldsymbol{S}_{t-1}.\,\mathrm{append}(\boldsymbol{k}_t, \boldsymbol{v}_t).\,\mathrm{drop}(\boldsymbol{k}_{t-M}, \boldsymbol{v}_{t-M})$	$\sum_{i=t-M}^{t} \exp(\boldsymbol{q}_t^\intercal \boldsymbol{k}_i)\,\|\boldsymbol{v} - \boldsymbol{v}_i\|^2$
Linear RNNs: Inductivist-based finite key-value associative memory registers		
LA [98]	$\boldsymbol{S}_t = \boldsymbol{S}_{t-1} + \boldsymbol{v}_t\boldsymbol{k}_t^\intercal$	$-\langle \boldsymbol{S}_t\boldsymbol{k}_t, \boldsymbol{v}_t\rangle$
GLA [107]	$\boldsymbol{S}_t = \boldsymbol{S}_{t-1}\mathrm{diag}(\boldsymbol{\alpha}_t) + \boldsymbol{v}_t\boldsymbol{k}_t^\intercal$	$-\langle \boldsymbol{S}_t\boldsymbol{k}_t, \boldsymbol{v}_t\rangle + \frac{1}{2}\left\|\sqrt{\mathrm{diag}(\boldsymbol{1} - \boldsymbol{\alpha}_t)}\boldsymbol{S}_t\right\|_F^2$
HGRN2 [79]	$\boldsymbol{S}_t = \boldsymbol{S}_{t-1}\mathrm{diag}(\boldsymbol{\alpha}_t) + \boldsymbol{v}_t(\boldsymbol{1} - \boldsymbol{\alpha}_t)^\intercal$	$-\langle \boldsymbol{S}_t(\boldsymbol{1} - \boldsymbol{\alpha}_t), \boldsymbol{v}_t\rangle + \frac{1}{2}\left\|\sqrt{\mathrm{diag}(\boldsymbol{1} - \boldsymbol{\alpha}_t)}\boldsymbol{S}_t\right\|_2^2$
Mamba2 [26]	$\boldsymbol{S}_t = \alpha_t^{\sim 1}\boldsymbol{S}_{t-1} + \beta_t^{\sim 0}\boldsymbol{v}_t\boldsymbol{k}_t^\intercal$	$-\beta_t\,\langle \boldsymbol{S}_t\boldsymbol{k}_t, \boldsymbol{v}_t\rangle + \frac{1}{2}\left\|\sqrt{1 - \alpha_t}\boldsymbol{S}_t\right\|_2^2$
Bilinear RNNs: Moving beyond linear key-value memory registers with memory correction		
G-DeltaNet [105]	$\boldsymbol{S}_t = \boldsymbol{S}_{t-1}\alpha_t^{\sim 1}\left(\boldsymbol{I}_t - \beta_t\boldsymbol{k}_t\boldsymbol{k}_t^\intercal\right) + \beta_t\boldsymbol{v}_t\boldsymbol{k}_t^\intercal$	$\frac{1}{2}\alpha_t\beta_t\left\|\frac{1}{\alpha_t}\boldsymbol{v}_t - \boldsymbol{S}_t\boldsymbol{k}_t\right\|^2 + \frac{1}{2}\left\|\sqrt{\boldsymbol{1} - \alpha_t}\boldsymbol{S}_t\right\|_2^2$
RWKV7 [73]	$\boldsymbol{S}_t = \boldsymbol{S}_{t-1}(\mathrm{diag}(\boldsymbol{\alpha}_t) - \beta_t\hat{\boldsymbol{k}}_t\hat{\boldsymbol{k}}_t^\intercal) + \boldsymbol{v}_t\tilde{\boldsymbol{k}}_t^\intercal$	$\frac{1}{2}\beta_t\left\|\frac{1}{\beta_t}\boldsymbol{v}_t - \boldsymbol{S}_t\boldsymbol{k}_t\right\|^2 + \frac{1}{2}\left\|\sqrt{\mathrm{diag}(\boldsymbol{1} - \boldsymbol{\alpha}_t)}\boldsymbol{S}_t\right\|_2^2$
Comba (ours)	$\boldsymbol{S}_t = \boldsymbol{S}_{t-1}\left(\alpha_t^{\sim 1} - \beta_t^{\downarrow}\boldsymbol{k}_t\boldsymbol{k}_t^\intercal\right) + \beta_t^{\uparrow}\boldsymbol{v}_t\boldsymbol{k}_t^\intercal$	$\frac{1}{2}\beta_t\,\|\boldsymbol{v}_t - \boldsymbol{S}_t\boldsymbol{k}_t\|^2 + \frac{1}{2}\left\|\sqrt{\boldsymbol{1} - \alpha_t}\boldsymbol{S}_t\right\|_2^2 - \langle \boldsymbol{q}_t, d\boldsymbol{k}_t\rangle$
(Modern) Nonlinear RNNs: Stronger expressiveness and memory capacity, but limited in chunk-wise parallelism.		
TTT-MLP [90]	$\boldsymbol{S}_t(\cdot) = \boldsymbol{S}_{t-B}(\cdot) - \sum_{i=1}^{B}\beta_i\nabla_S\,\|\psi(\boldsymbol{S}_{t-B}(\boldsymbol{k}_i)) - \boldsymbol{v}_i\|^2$	$\beta_i\,\|\boldsymbol{v}_i - \psi(\boldsymbol{S}_j(\boldsymbol{k}_i))\|^2$
MIRAS [6]	$\boldsymbol{S}_t = \alpha_t\boldsymbol{S}_{t-1} - \beta_t\nabla_S\,\|g(\psi(\boldsymbol{S}_{t-1}), \boldsymbol{k}_t) - \boldsymbol{v}_t\|_p^p$	$\beta_t\,\|\boldsymbol{v}_t - g(\psi(\boldsymbol{S}_t), \boldsymbol{k}_t)\|_p^p + \frac{1}{2}\left\|\sqrt{\mathrm{diag}(\boldsymbol{1} - \boldsymbol{\alpha}_t)}\boldsymbol{S}_t\right\|_2^2$

$\boldsymbol{S}_t$ denotes memory, while $\boldsymbol{k}_t, \boldsymbol{v}_t$ are key-value pairs. $\alpha_t^{\sim 1}$ denotes close to 1. $\beta_t^{\downarrow}, \beta_t^{\uparrow}$ represent smaller/bigger factors. Early Nonlinear RNNs, such as LSTM [46] and GRU [21] are omitted on the table. Softmax attention can also be interpreted as an L1 loss with a kernel function [114].

The data-dependent gating in Linear RNNs provides a dynamic memory management similar to the adaptive, weighted information fusion in softmax attention. This allows such models to selectively update and retain relevant information, enabling Linear RNNs to serve as practical replacements for Transformers in many downstream tasks [48, 62, 49]. However, this mechanism remains heuristic; that is, the model lacks a criterion for determining which memories to forget, and all key-value associations are forgotten uniformly, rendering the process less targeted and efficient [105].

To address this, recent models such as DeltaNet [108, 105], RWKV-7 [73], and TTT [90] have advanced state transition to generalized Householder transformations [51], enhancing model's learning capacity and enabling supervised memory control via the Delta learning rule [101]. These architectural shifts foster richer interactions between the internal state $\boldsymbol{S}$ and the input information $\boldsymbol{k}$, moving beyond simple linear *key-value memory registers* and resembling bilinear dynamics. Consequently, we refer to these models as Bilinear RNNs in this paper. Further improvements [7, 52, 6] have built upon TTT by introducing higher-order nonlinear optimization or MLP-based deep memory, giving rise to modern Nonlinear RNNs, which enhance expressiveness but sacrifice the ability to perform chunk-wise parallelism over the sequence. In summary, research in this field is still in its early stages, and there remain open challenges in achieving a good balance between model expressiveness and hardware-efficient implementation during pretraining.

In this paper, we make the following contributions:

- We summarize the progress of efficient sequence modeling methods in Table 1, and highlight the core design principles behind recent advances within the concept of Bilinear RNNs. (Section 2)
- Inspired by closed-loop control theory, we propose a novel Bilinear RNN architecture named ***Comba***. Unlike previous models, Comba features a scalar-plus-low-rank (SPLR) state transition and applies feedback control to the query vector during the output. We further develop a hardware-

friendly implementation of Comba using chunk-wise parallelism in Triton [93], which achieves a 40% speed improvement in forward propagation compared to Gated-DeltaNet. (Section 3)

- We pretrain models with 340M and 1.3B parameters and evaluate their performance across a range of tasks, including language modeling and vision tasks. Extensive ablation studies are conducted to assess the impact of key components of Comba. (Section 4)

2 Preliminary & Related Works

2.1 Linear RNNs

Unlike autoregressive Transformers that store all contextual information in KV cache, linear RNNs compress highly abstract knowledge into a fixed-size state for generalization, structurally resembling energy-based models [56], i.e., Hopfield networks [65, 31] and neural Hebbian learning systems [18]. Early models like Linformer [98], S4 [41] and RetNet [91] lack effective, data-dependent memory control, resulting in inferior performance to softmax attention. Later models like Mamba [38] and GLA [107] address this by introducing a dynamic, projection-based gating, yielding substantial improvements. Formally, these models are linear register systems with *key-value associative memory*, where the memory is written by the forgetting/input gates ($\boldsymbol{\alpha}$, $\boldsymbol{\beta}$) and retrieved via query-based read:

$$\boldsymbol{S}_t = (\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t) @ (\boldsymbol{S}_{t-1}, \boldsymbol{k}_t^\intercal \boldsymbol{v}_t)^\intercal \quad \textit{(Write)}, \qquad \boldsymbol{o}_t = \boldsymbol{S}_t \boldsymbol{q}_t \quad \textit{(Read)}. \tag{1}$$

Due to differences in theoretical foundations and development trajectories, linear RNNs have evolved into two main implementation paradigms: (i) linear attentions (LAs) [112, 79, 22, 107, 105, 5] and (ii) state space models (SSMs) [72, 38, 39], we summarize their key distinctions in Table 2:

i) ***State size***: SSMs like Mamba2 [26] adopt a fixed state expand dimension of 128, resulting in a state size of 256D. Whereas in LAs, the state size is determined by the dimension of key/value heads (typically 64). In practice, Mamba2 offers a higher capacity [20], which underpins its advantages in retrieval tasks and hybrid architectures [88, 89].

Table 2: Comparison between SSM and LA families with the head number H.

Component	SSMs (Mamba2)	LAs
Input value	$\boldsymbol{u}_t \in \mathbb{R}^{2D/H}$	$\boldsymbol{v}_t \in \mathbb{R}^{dv/H}$
State expand	$\bar{\boldsymbol{B}}_t \in \mathbb{R}^{128}$	$\boldsymbol{k}_t \in \mathbb{R}^{dk/H}$
Output	$\bar{\boldsymbol{C}}_t \in \mathbb{R}^{128}$	$\boldsymbol{q}_t \in \mathbb{R}^{dk/H}$
State size	$\boldsymbol{x}_t : 256 \times D$	$\boldsymbol{S}_t : \frac{dk \times dv}{H}$
MLP	✘	✔
Mode	Multi-value	Multi-head

ii) ***Parameter composition***: SSMs like Mamba2 resembles a multi-value attention mechanism [26], where the input projection $\bar{\boldsymbol{B}}$ ($\boldsymbol{k}$) and output projection $\bar{\boldsymbol{C}}$ ($\boldsymbol{q}$) are shared across all value heads $\boldsymbol{u}$ ($\boldsymbol{v}$). Additionally, state space models omit the feedforward network and instead double both the input dimension and the number of layers to increase model capacity. In this paper, we empirically follow the linear attention design but keep the head dimension at 256 to match the state size of Mamba2. A detailed architectural ablation is in §4.2.

Chunk-wise Parallel Although Linear RNNs achieve a favorable pretraining time complexity of $\mathcal{O}(LD^2)$, they often slower than softmax attention with $\mathcal{O}(L^2D)$ complexity on shorter sequences. This is mainly due to the fact that current hardware is highly optimized for `matmul` operations, which limits the efficiency of linear recurrence, necessitating additional training optimizations. S4 [41] introduces a complex diagonal-plus-low-rank design using the Cauchy kernel, which is later simplified in DSS [43] and S4D [40] through complex and real diagonal approximations to employ Vandermonde-based convolutions. Other models like S5 [84] and Mamba [38] leverage the Blelloch scan algorithm [15] to cache intermediate results and speed up recurrent computation. Recent methods inspired by FlashAttention [25], including Lightning-Attns [77, 78, 58], GLA [107], and Mamba2 [26], introduce inter-chunk recurrence combined with intra-chunk parallelism to fully utilize matrix compute throughput. A basic formulation using chunk size C can be expressed as:

$$\boldsymbol{S}_{[t+1]} = \boldsymbol{S}_{[t]} + \boldsymbol{V}_{[t]}^\intercal \boldsymbol{K}_{[t]} \ \in \mathbb{R}^{D \times D}, \qquad \boldsymbol{O}_{[t]} = \boldsymbol{Q}_{[t]} \boldsymbol{S}_{[t]}^\intercal + (\boldsymbol{Q}_{[t]} \boldsymbol{K}_{[t]}^\intercal \odot \mathrm{Mask}_{[t]}) \boldsymbol{V}_{[t]} \ \in \mathbb{R}^{C \times D}. \tag{2}$$

2.2 Bilinear RNNs and Beyond

In a neural memory perspective [34], effective memory management remains a central challenge. Unlike Hebbian learning rule [18], which relies on reinforcement-based memory updates, Delta learning rule [75, 101] focuses on supervised memory control and has been extensively explored in various works, such as fast weight programs [50, 81] and Meta-learning [68, 67]. (Gated-)DeltaNet

[108, 105] employs it as a memory correction $\boldsymbol{v}_t^{\text{new}} = \boldsymbol{v}_t - \boldsymbol{S}_{t-1}\boldsymbol{k}_t$, and based on Eq. 2, introduces an efficient chunk-wise parallel algorithm for hardware-efficient training in sequence modeling:

$$\boldsymbol{S}_t = \boldsymbol{S}_{t-1} - \beta_t(\boldsymbol{S}_{t-1}\boldsymbol{k}_t - \boldsymbol{v}_t)\boldsymbol{k}_t^{\intercal} = \boldsymbol{S}_{t-1}(\boldsymbol{I} - \beta_t\boldsymbol{k}_t\boldsymbol{k}_t^{\intercal}) + \beta_t\boldsymbol{v}_t\boldsymbol{k}_t^{\intercal}, \qquad \boldsymbol{o}_t = \boldsymbol{S}_t\boldsymbol{q}_t. \tag{3}$$

These models are no longer a linear *key-value memory register* as in Eq. 1; instead, the interaction between the state $\boldsymbol{S}$ and the input $\boldsymbol{k}$ introduces a bilinear term $\boldsymbol{Sk}$, this resembles a affine bilinear system[1] [17, 113, 99, 70]. Accordingly, models that are similar to the updating rule in Eq. 3 can be classified as ***Bilinear RNNs***, and their key strengths are summarized as follows:

Supervised Memory Management As shown in Fig. 1, the Householder transform $(\boldsymbol{I} - \beta_t\boldsymbol{k}_t\boldsymbol{k}_t^{\intercal})$ generated by the Delta rule defines a mirror transform, effectively reflecting stored memories across a hyper-plane orthogonal to $\boldsymbol{k}_t$. Given a memory $\boldsymbol{S} \in \mathbb{R}^{D\times D}$ with at most D orthogonal memory patterns, when the sequence length $t > D$, to avoid memory conflicts, this reflection attenuates components of stored memories $\{\boldsymbol{S}_{t-1}^i\}_{i=1}^D$ in non-orthogonal directions based on factor β_t. This mechanism implicitly enforces orthogonal memory management, enabling the model to preserve D distinguishable historical memories over time. In this paper, Comba modifies the state transition to $(\alpha_t - \beta_t\boldsymbol{k}\boldsymbol{k}^{\intercal})$, which is a scalar-plus-low-rank form (SPLR), enabling more flexible supervision. To some extent, this process can be seen as a Schmidt orthogonalization [57] or a rotation operation [86] on the KV cache.

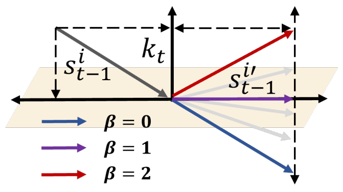

Figure 1: Householder transform as mirror transform with factor β.

Richer Expressive Power Linear RNNs generally approximate the dense state transition matrix [66, 46] with a diagonal matrix or scalar [40, 26], significantly reducing computational overhead but at the cost of expressiveness [1]. While the additional low-rank terms of the state transition in Eq. 3 improve the model's expressiveness while preserving tractability for efficient parallelization.

Building on this, Gated-DeltaNet [105] introduces a global scalar forgetting gate on the state, and Delta-Product [83] explore a multi-step Householder transform $\prod_{j=1}^{n}(\boldsymbol{I} - \beta_{t,j}\boldsymbol{k}_{t,j}\boldsymbol{k}_{t,j}^{\top})$, enabling a smooth interpolation between purely diagonal and fully dense transitions. RWKV7 [73] improves the IPLR form to diagonal-plus-low-rank (DPLR) form, which aligns with insights from HiPPO theory [39, 42], which shows that all orthogonal polynomial projection matrices can be decomposed into DPLR components. Our proposed Comba adopts a scalar-plus-low-rank (SPLR) form. Empirically, we find that the scalar is sufficiently expressive (similar to the empirical simplification from Mamba1 to Mamba2) and offers significant pretraining acceleration over RWKV-7.

From another perspective, models such as TTT [90] separate the model weights into inner and outer components, updating the inner memory directly via stochastic gradient descent (SGD) [16].

$$\boldsymbol{S}_t = \boldsymbol{S}_{t-B} - \sum_{i=1}^{B}\beta_i\nabla_S\left\|\text{LayerNorm}(\boldsymbol{S}_{t-B}\boldsymbol{k}_i) - \boldsymbol{v}_i\right\|^2, \qquad \boldsymbol{o}_t = \boldsymbol{S}_t\boldsymbol{q}_t, \tag{4}$$

where the state $\boldsymbol{S}$ can be parameterized by a matrix or a two-layer MLP to increase memory capacity. When mini-batch $B = 1$ and ignoring normalization operation, this reduces to the update rule in Eq. 3, and when adopting MLP-based deep memory, the model transitions into a modern form of Nonlinear RNN. Subsequently, Titans [7] introduces a data-dependent state decay and momentum, while models like MIRAS [6] and Lattice [52] enhance $\boldsymbol{S}$ using nonlinear higher-order optimization, e.g., sign function or high-order derivatives. However, there is no free lunch: these models rely on mini-batch gradient descent approach to compensate for the sequence-level parallelism barrier introduced by their structural complexity, but empirical results show that a mini-batch of 1 (similar to Eq. 3, which is a first-order approach) remains optimal [90], especially in language modeling. Comba retains the bilinear form and offers an efficient chunk-wise parallel optimization.

3 Bilinear RNNs in Closed-loop Control

Unlike perspectives from neural memory or optimization, this work revisits Bilinear RNNs through the lens of control theory [23, 24, 87]. As shown in Table 3, linear RNNs such as Mamba2 and

[1]Bilinear systems are linear with respect to state and input individually, but nonlinear overall due to the product term (e.g., $\boldsymbol{Sk}$). They are regarded as a special class of nonlinear systems that preserve controllability.

Table 3: Update rules in a control/neural memory perspective, with feedback $\boldsymbol{P}(\cdot)$ and scalar factor d.

Option	Open-loop Control (Mamba2)	Close-loop Control (Comba)	Gated Delta Rule
Input / Memorize	$\boldsymbol{S}_t = \alpha_t \boldsymbol{S}_{t-1} + \beta_t \boldsymbol{v}_t^{\text{new}} \boldsymbol{k}_t^\intercal$	$\boldsymbol{S}_t = \alpha_t \boldsymbol{S}_{t-1} + \beta_t \boldsymbol{v}_t^{\text{new}} \boldsymbol{k}_t^\intercal$	$\boldsymbol{S}_t = \alpha_t \boldsymbol{S}_{t-1} + \beta_t \boldsymbol{v}_t^{\text{new}} \boldsymbol{k}_t^\intercal$
Feedback / Reflect	N/A	$\boldsymbol{v}_t^{\text{new}} = \boldsymbol{v}_t - \boldsymbol{P}_t(\boldsymbol{S}_{t-1})$	$\boldsymbol{v}_t^{\text{new}} = \boldsymbol{v}_t - \alpha_t \boldsymbol{S}_{t-1} \boldsymbol{k}_t$
Output / Recollect	$\boldsymbol{o}_t = \boldsymbol{S}_t \boldsymbol{q}_t$	$\boldsymbol{o}_t = \boldsymbol{S}_t \boldsymbol{q}_t - d\boldsymbol{P}_t(\boldsymbol{S}_t)$	$\boldsymbol{o}_t = \boldsymbol{S}_t \boldsymbol{q}_t$

GLA are generally viewed, to some extent, as open-loop control systems, where the output does not provide feedback to influence the control behavior. In contrast, another class of systems, known as closed-loop control systems [45], incorporates negative feedback to enhance the adaptability of the system and allows it to handle more complex dynamic tasks. According to Wikipedia, a closed-loop controller should use feedback to control states or outputs of a dynamical system. So in this paper, Comba adopts a two-stage feedback strategy: the input information $\boldsymbol{v}_t$ is first corrected via state-based feedback $\boldsymbol{P}_t(\cdot)$, and the output is similarly refined using the same feedback mechanism. Compared to directly output correction methods, e.g., use $\boldsymbol{q}_t$ to compute $\boldsymbol{v}_t^{\text{new}}$, this approach is generally considered more robust and better suited for parallel computation (as it only modifies $\boldsymbol{q}_t$ at the output stage without involving it in state updates). From this perspective, models such as TTT, DeltaNet, and RWKV-7 incorporate only the first-stage state feedback correction.

Feedback Parameterization Similar to the optimizer perspective, the feedback $\boldsymbol{P}_t(\cdot)$ in closed-loop control can be either linear and first-order, or nonlinear and higher-order. However, two major challenges arise: (i) adopting nonlinear or high-order optimization techniques, as in TTT, Lattice, or MIRAS, will hinder chunk-wise parallelism; and (ii) recurrent models suffer from the well-known issue of exponential gradient explosion [74] during training. If we follow DeltaNet using first-order feedback but initialize a new vector [106] to interact with the state, it becomes difficult to ensure that the spectral radius of the state transition matrix remains below one. Therefore, considering these factors, Comba follows previous models using the $\boldsymbol{k}$ vector to interact with the state $\boldsymbol{S}$, while introducing a special treatment of feedback strength detailed in the following sections. For a standard Comba with forget gate α_t, state feedback factor $\tilde{\beta}_t$, input gate β_t, and output feedback factor d:

$$\boldsymbol{S}_t = \underbrace{\boldsymbol{S}_{t-1}(\alpha_t - \tilde{\beta}_t \boldsymbol{k}_t \boldsymbol{k}_t^\intercal)}_{\text{State correction}} + \beta_t \boldsymbol{v}_t \boldsymbol{k}_t^\intercal \quad \in \mathbb{R}^{dv \times dk}, \qquad \boldsymbol{o}_t = \underbrace{\boldsymbol{S}_t(\boldsymbol{q}_t - d\boldsymbol{k}_t)}_{\text{Output correction}} \quad \in \mathbb{R}^{dv} \tag{5}$$

Compared to previous Bilinear RNNs, Comba exhibits the following structural differences:

Scalar Plus Low-Rank (SPLR) In practice, the scalar gating will be limited to the interval $(0, 1)$, and $\|\boldsymbol{k}_t\| = 1$ (L2 Norm). DeltaNet formulates the state transition in an IPLR structure, while RWKV-7 improves expressiveness via a DPLR extension with LoRA-style [47] diagonal initialization, i.e., $\text{diag}(\boldsymbol{w}_t) - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\intercal$. However, our results indicate that the low-rank form will impair model capacity. Moreover, recent efforts [12, 55, 11, 69] aim to distill Transformers into recurrent structures to leverage prior knowledge of large-scale pretrained Transformers, where a key design principle is to minimize additional parameters. To this end, Comba adopts an SPLR structure that only introduces a data-dependent scalar, achieving superior empirical performance compared to RWKV-7 (as implemented in `FLA` [109]), along with a 2 × pretraining acceleration (the memory usage is only half). Recent works [83, 36] have also improved IPLR structures by extending their eigenvalues into the negative domain, e.g., $\boldsymbol{I} - 2\beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\intercal$, to enhance the model's state-tracking capability. Notably, SPLR naturally admits negative eigenvalues. For fair comparison, in Table 4, we introduce a variant named Comba-iplr. However, such models with global state decay tend to overfit, and the SPLR structure remains optimal and is more structurally aligned with control theory.

Table 4: State Transition for Comba Variants.

Version	State transition	Eigenvalues
Comba-iplr	$\alpha_t(\boldsymbol{I} - 2\tilde{\beta}_t \boldsymbol{k}_t \boldsymbol{k}_t^\intercal)$	$(-1, 1)$
Comba-splr	$(\alpha_t - \tilde{\beta}_t \boldsymbol{k}_t \boldsymbol{k}_t^\intercal)$	$(-1^{\sim 0}, 1)$

Output Correction Comba introduces additional output feedback, from an optimization perspective, this is equivalent to incorporating a similarity optimization objective $\langle \boldsymbol{q}_t, d\boldsymbol{k}_t \rangle$ with factor d. In a neural memory perspective, $\boldsymbol{k}$ ensures that memory $\boldsymbol{v}$ is stored as clearly as possible, enabling precise querying by $\boldsymbol{q}$. This optimization objective directly facilitates this process, and significantly reduces the model's perplexity, thereby enhancing performance (§4.2). Empirically, initializing d

to 0.02 improves performance for some smaller models (e.g., 340M), enabling gradual learning of the similarity between $\boldsymbol{q}$ and $\boldsymbol{k}$. For larger models (e.g., 1.3B, 2.7B), initializing d to 1 leads to the greatest performance[2]. While prior research has largely focused on improving gating mechanisms or optimizing state updates via key-value operations, few models have explicitly modified the output pathway (i.e., the query @ State). We think there appears to be a potential connection between Comba and MesaNet [96], as MesaNet also performs query correction in the output stage by solving a closed-form recursive least squares problem [10], resulting in $\boldsymbol{q}_t = (\boldsymbol{H}_t + \boldsymbol{\Lambda})^{-1} \boldsymbol{q}_t$. Notably, the correction in Comba can be implemented with a single line of code $\boldsymbol{q}_t - d\boldsymbol{k}_t$ and is applicable to nearly all RNN variants. In App. A.2, we also provide an alternative explanation, namely that d is equivalent to the residual matrix $\boldsymbol{D}$ in Mamba [38].

Table 5: Various initialization examples and numerical range of the existing recurrent model gates.

Model	**Forget Gate $\boldsymbol{\alpha}_t$**	**Range**	**Input Gate $\boldsymbol{\beta}_t$**	**Range**
GLA [107]	$\text{sigmoid}(\boldsymbol{W}_1\boldsymbol{W}_2\boldsymbol{x}_t)^{\frac{1}{\tau}}\mathbf{1}^\intercal$	$(0, 1)$	N/A	1
Mamba2 [26]	$\exp\left(-a\,\text{softplus}\left(\boldsymbol{W}_\alpha\boldsymbol{x}_t + c\right)\right)$	~ 1	$\text{softplus}\left(\boldsymbol{W}_\alpha\boldsymbol{x}_t + c\right)$	~ 0
MetaLA [22]	$\text{sigmoid}(\boldsymbol{W}_\alpha\boldsymbol{x}_t)\mathbf{1}^\intercal$	$(0, 1)$	$\mathbf{1} - \text{sigmoid}(\boldsymbol{W}_\alpha\boldsymbol{x}_t)$	$(0, 1)$
Comba (**ours**)	$\exp\left(-a\,\text{softplus}\left(\boldsymbol{W}_\alpha\boldsymbol{x}_t + c\right)\right)$	~ 1	$\text{sigmoid}(\boldsymbol{W}_\beta\boldsymbol{x}_t)$	$\tilde{\beta}_t < \beta_t \in (0, 1)$

3.1 Forcing Forgetting for Long-range Modeling

Extensive work [61, 92, 76] has shown that positional encoding is crucial for language models to generalize from short pretraining contexts to unconstrained inference lengths. While gated linear recurrent structures themselves can be viewed as the continuous accumulation of bias [92], enabling these models to extrapolate effectively by learning how to forget. Recent studies [20, 110, 9] suggest that pretraining lengths (e.g., 2K or 4K tokens) are insufficient to fill the model's state capacity, thus making the forget gate close to 1, forcing the model to learn how to forget. As shown in Table 5, we evaluate representative gating initialization methods. For models like GLA [107], the forget gate is initialized to $(0, 1)$ and retains all incremental memories. For MetaLA [22], HGRN2 [79], and GSA [112], the sum of the forget and input gates is constrained to 1, achieving a relative balance in system memory capacity. Experimental results in §4.2 indicate that breaking this balance is necessary. To this end, Comba follows Mamba2's design by constraining the forget gate α_t to close to 1, while separately initializing the input gate β_t to $(0, 1)$ [108]. Additionally, we set the strength of state feedback correction is $\tilde{\beta}_t = b \odot \beta_t$, where b is computed by Sigmoid function to be constrained in interval $(0, 1)$ to ensure that the feedback strength is weaker than the incremental information.

3.2 Comba with Chunk-wise Parallel

Based on Eq. 5, Comba can be implemented recursively, enabling constant memory usage and $\mathcal{O}(1)$ time complexity during the inference stage, where the Python-style pseudocode is provided in App. B.1. However, the naive recursive implementation in PyTorch lacks sufficient matrix multiplication and thread parallelism [25], resulting in unacceptable overhead to pretraining. Referring to DeltaNet [105], we optimize Comba through chunk-wise parallelism. App. B.3 presents an alternative form to fuse feedback decay factor b into $\boldsymbol{k}$ as $\boldsymbol{p} = b\boldsymbol{k}$ for a flexible invocation.

In the following illustration, $\square_{[t]} \in \mathbb{R}^{C\times d}$ for $\square \in \{\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}, \boldsymbol{O}, \boldsymbol{U}, \boldsymbol{W}\}$ defines the chunkwise matrices that stack the $\boldsymbol{q}_t, \boldsymbol{k}_t, \boldsymbol{v}_t, \boldsymbol{o}_t, \boldsymbol{u}_t, \boldsymbol{w}_t$ vectors. Additionally, we set $\square_{[t]}^{1:r} = \prod_{i=tC}^{tC+r} \square_{[t]}^{i}$, $\text{Diag}(\square_{[t]}^{1\to r}) = \text{Diag}\{\square_{[t]}^{1}, \ldots, \square_{[t]}^{r}\}$, and $\mathcal{A}_{[t]}^{i/j} \in \mathbb{R}^{C\times C}$ is a matrix with element $\alpha_{[t]}^{1:i}/\alpha_{[t]}^{1:j}$.

By partially expanding the recurrence to a chunk-wise formulation for Eq. 5, we have:

$$\boldsymbol{S}_{[t]}^{r} = \boldsymbol{S}_{[t]}^{0} \underbrace{\left(\prod_{i=1}^{r}\left(\alpha_{[t]}^{i} - \tilde{\beta}_{[t]}^{i}\boldsymbol{k}_{[t]}^{i}\boldsymbol{k}_{[t]}^{i\intercal}\right)\right)}_{:=\boldsymbol{D}_{[t]}^{r}\ (\text{“pseudo” memory decay})} + \underbrace{\sum_{i=1}^{r}\left(\beta_{[t]}^{i}\boldsymbol{v}_{[t]}^{i}\boldsymbol{k}_{[t]}^{i\intercal}\prod_{j=i+1}^{r}\left(\alpha_{[t]}^{j} - \tilde{\beta}_{[t]}^{j}\boldsymbol{k}_{[t]}^{j}\boldsymbol{k}_{[t]}^{j\intercal}\right)\right)}_{:=\boldsymbol{H}_{[t]}^{r}\ (\text{“pseudo” Incremental memory})} \tag{6}$$

[2]We find that $d = 1$ yields better performance in most cases, which is the default configuration for Comba.

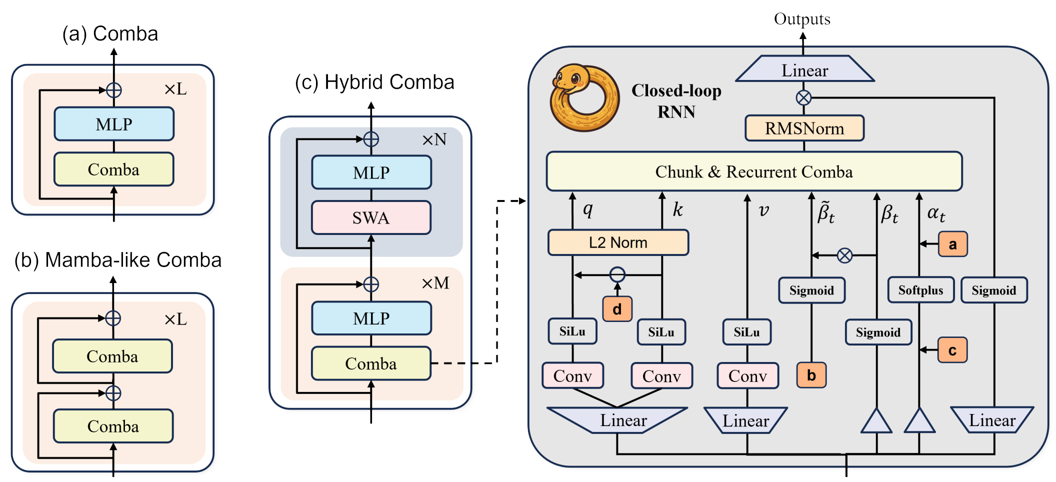

Figure 2: **Comba Families.** The Mamba-like architecture omits MLP layers, uses multi-value attention, and doubles the model depth. For the hybrid model, we incorporate sliding window attention in flexible proportions to boost the model's recall ability. The window size is set to the context length, equivalent to softmax attention.

Eq. 6 involves matrix-matrix products at each time step, i.e., $\boldsymbol{S}^0_{[t]} @ \boldsymbol{D}^r_{[t]}$, preventing parallelization in the sequence level. Then, we employ the WY representation [13] to eliminate these terms:

$$\boldsymbol{D}^r_{[t]} = \alpha^{1:r}_{[t]} - \sum_{i=1}^{r} \alpha^{i:r}_{[t]} \boldsymbol{w}^i_{[t]} \boldsymbol{k}^{i\intercal}_{[t]}, \qquad \boldsymbol{w}^r_{[t]} = \tilde{\beta}^r_{[t]} \left(\alpha^{1:r-1}_{[t]} \boldsymbol{k}^r_{[t]} - \sum_{i=1}^{r-1} \boldsymbol{w}^i_{[t]} \left(\alpha^{i:r-1}_{[t]} \boldsymbol{k}^{i\intercal}_{[t]} \boldsymbol{k}^r_{[t]} \right) \right) \tag{7}$$

$$\boldsymbol{H}^r_{[t]} = \sum_{i=1}^{r} \alpha^{i:r}_{[t]} \boldsymbol{u}^i_{[t]} \boldsymbol{k}^{i\intercal}_{[t]}, \qquad \boldsymbol{u}^r_{[t]} = \beta^r_{[t]} \boldsymbol{v}^r_{[t]} - \tilde{\beta}^r_{[t]} \sum_{i=1}^{r-1} \boldsymbol{u}^i_{[t]} \left(\alpha^{i:r-1}_{[t]} \boldsymbol{k}^{i\intercal}_{[t]} \boldsymbol{k}^r_{[t]} \right) \tag{8}$$

To maximize hardware efficiency, we apply the UT transform [51] to Eq. 7-8 to reduce non-matmul FLOPs, which is crucial to enable better hardware utilization during training:

$$\boldsymbol{W}_{[t]} = \boldsymbol{M}_{[t]} \mathrm{Diag}\left(\tilde{\beta}^{1\to C}_{[t]} \odot \alpha^{0\to(C-1)}_{[t]} \right) \boldsymbol{K}_{[t]}, \qquad \boldsymbol{U}_{[t]} = \boldsymbol{M}_{[t]} \mathrm{Diag}\left(\beta^{1\to C}_{[t]} \right) \boldsymbol{V}_{[t]} \tag{9}$$

$$\boldsymbol{M}_{[t]} = \left(\boldsymbol{I} + \mathrm{lower}\left(\mathrm{Diag}\left(\tilde{\beta}^{1\to C}_{[t]} \right) \left(\mathcal{A}^{(i-1)/j}_{[t]} \odot \boldsymbol{K}_{[t]} \boldsymbol{K}^\intercal_{[t]} \right) \right) \right)^{-1} \tag{10}$$

The inverse of a lower triangular matrix can be efficiently computed through an iterative row-wise approach by forward substitution in Gaussian elimination [37] and maintain data in float32. Notably, Comba computes the inverse matrix once in Eq. 10 (twice in the original Gated-DeltaNet). This is mainly attributed to the introduction of a new form of mathematical induction in deriving the WY representation in Eq. 7, leading to a more concise formulation and resulting in speedup[3].

Finally, we can formulate Eq. 5 in a matrix form to perform chunk-wise parallel training:

$$\boldsymbol{S}_{[t+1]} = \alpha^{1:C}_{[t]} \boldsymbol{S}_{[t]} + \left(\boldsymbol{U}_{[t]} - \boldsymbol{W}_{[t]} \boldsymbol{S}^\intercal_{[t]} \right)^\intercal \mathrm{Diag}\left(\alpha^{i\to C}_{[t]} \right) \boldsymbol{K}_{[t]} \tag{11}$$

$$\boldsymbol{O}_{[t]} = \underbrace{\mathrm{Diag}\left(\alpha^{1\to C}_{[t]} \right) \tilde{\boldsymbol{Q}}_{[t]} \boldsymbol{S}^\intercal_{[t]}}_{\text{inner chunk}} + \underbrace{\mathrm{Tril}(\tilde{\boldsymbol{Q}}_{[t]} \boldsymbol{K}^\intercal_{[t]} \odot \mathcal{A}^{i/j}_{[t]})}_{\text{intra chunk}} \underbrace{\left(\boldsymbol{U}_{[t]} - \boldsymbol{W}_{[t]} \boldsymbol{S}^\intercal_{[t]} \right)}_{\text{“pseudo”-value term}} \tag{12}$$

where the query matrix $\tilde{\boldsymbol{Q}}_{[t]}$ is also influenced by the feedback control and can be precomputed by: $\tilde{\boldsymbol{Q}}_{[t]} = \boldsymbol{Q}_{[t]} - \mathrm{Diag}(d^{1\to C}_{[t]}) \boldsymbol{K}_{[t]}$ in chunk-wise at minimal cost.

[3]During the forward pass, Comba alleviates the main bottleneck of inverse matrix computation with $\mathcal{O}(d^3)$ in the original Gated-DeltaNet, yielding a 40% speedup. In the backward pass, performance gains diminish due to the reuse of cached inverse matrices $\boldsymbol{M}$. However, in large-scale models where recomputation is required, Comba is expected to offer significant performance advantages. Moreover, this structured modification applies to almost all models with Delta rule. Recently, inspired by Comba, Gated-DeltaNet has also been upgraded to a single inversion in `flash-linear-attention`, resulting in a significant speedup.

3.3 Neural Architecture

As shown in Fig. 2, we present the architecture for Comba families, where $\{a, b, c, d\}$ are trainable scalars (experimental results indicate that data dependency is not required). Following prior work [22, 105, 38, 32], we introduce short convolutions to $\boldsymbol{qkv}$ to incorporate token shift to improve the model's retrieval capacity. We also retain feature map operations [98] and utilize the SiLU function to approximate the exponential kernel in the softmax attention. To further stabilize training, we apply L2 normalization to $\boldsymbol{qk}$ and employ a $\mathrm{Sigmoid}$-based gating mechanism. Additionally, we explore a hybrid architecture [89, 105, 80] by integrating Comba layers directly with softmax attention.

4 Experiments

Setting In this paper, all models are pretrained based on `flash-linear-attention` [109] repository and utilize *NVIDIA A800-80G GPUs*. The 340M Comba pretraining requires *8×10 GPU hours*, while the 1.3B Comba requires *32×48 GPU hours*. We employ the AdamW optimizer [63] with a 3e-4 learning rate, cosine schedule, 0.01 weight decay, and 1.0 gradient clipping. Random seed is 42.

4.1 Operator Efficiency Analysis

As shown in Fig. 3, we compared the speeds of various operators in both forward and backward processes. The recurrent Comba in PyTorch [71] incurs significant computational overhead, limiting its scalability for large-scale pretraining. Flash-attention [25] achieves the fastest speed for shorter sequences (e.g., 1024), but its quadratic complexity results in decreasing efficiency as sequence length increases. Among four modern RNN operators, Comba shows nearly 40% speed improvement in the forward process over Gated-DeltaNet due to a more efficient formula structure. GLA suffers from slower operator speed due to the use of diagonal gating matrices. Although RetNet achieves the fastest speed, it falls behind other models in performance due to the lack of data-dependent memory management (Table 6). Overall, Comba shows potential as a foundational framework.

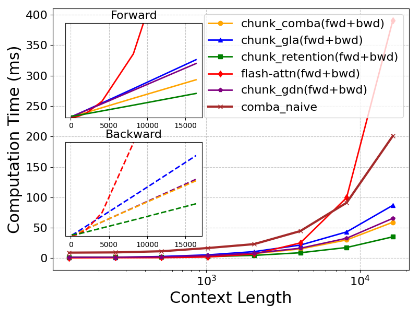

Figure 3: Operator speed evaluated on the `Triton-Testing-Benchmark` [93] (fwd and bwd) in single A800-80G GPU.

4.2 Language Modeling & Architecture Ablation

Commonsense Reasoning Ability As shown in the left half of Table 6, (i) most recursive models outperform transformers in commonsense reasoning tasks, due to their recursive structure that resembles a chain of thought [100]. (ii) The SPLR structure outperforms both the IPLR and DPLR structures in two model sizes and achieves the highest computational efficiency. (iii) Output correction, i.e., $\langle q, dk\rangle$, significantly reduces perplexity, enhancing memory utilization during question answering and improving performance across various metrics. (iv) We find that the Mamba architecture design is suboptimal. MLP, as a special (non-Hebbian) key-value memory [35, 34], complements the key-value associative memory in the state, which is particularly important for tasks such as inference. (v) Although Mamba's multi-value attention model aids efficient key-value memory storage [59], it sacrifices performance compared to standard multi-head attention. (vi) We discover that the initialization of d should be chosen differently for model scales, e.g., $d = 0.02$ for 340M models and $d = 1$ for 1.3B models, as smaller models are more prone to incorrect gradient descent directions.

Figure 4 shows the training loss curves of various architectures, and Comba, particularly with output correction, demonstrates lower loss and greater expressive power. However, in our experiments, we found that the IPLR structure typically yields lower loss. This may be because (i) as shown in Table 4, the range for the SPLR structure eigenvalues is slightly smaller than that of IPLR, likely due to the special initialization of α. (ii) We observe that in visual modeling, the IPLR version of the model often experiences a rapid loss decrease followed by an increase. Combining these findings with the results in Table 6, we speculate that the IPLR structure is prone to overfitting.

Recall Ability & Hybrid Architecture As shown in the right half of Table 6, (i) the overall results align with the trends observed in commonsense reasoning tasks. (ii) Recursive models have traditionally struggled with limited recall due to their finite state space, unlike the transformer's unlimited

Table 6: Zero-shot performance of 340M and 1.3B models trained on `SlimPajama` [85] datasets. The commonsense Reasoning task is evaluated by `lm-evaluation-harness` [33] and the recall-intensive task follows `prefix-linear-attention` [3] with 2K input tokens. * Some of the baseline results are from [108] and [29].

Model & Scale	Lamb.†	Wiki.	ARC$_e$	ARC$_c$	Hella.	Lamb.	PIQA	Wino.	Avg.	FDA	SWDE	SQD.	NQ	TQA.	Drop	Avg.
	ppl$_\downarrow$	ppl$_\downarrow$	acc	acc$_n$	acc$_n$	acc	acc	acc	acc	acc	acc	acc	acc	acc	acc	acc
340M params with 15B training tokens and 0.5M batchsize tokens																
Trans++*	76.46	28.39	44.91	**25.94**	34.95	26.90	64.31	51.07	41.35	**46.14**	25.87	**33.22**	**18.94**	45.97	19.94	**31.68**
GLA	72.41	28.44	45.30	23.13	34.71	26.14	64.58	51.64	40.92	11.26	16.78	27.85	12.77	43.80	17.68	21.69
Mamba*	64.75	28.39	46.30	23.60	35.40	26.72	65.00	50.10	41.80	7.14	12.96	24.35	9.47	41.84	17.11	18.81
RWKV7	45.00	25.74	49.03	25.09	36.63	29.01	65.45	51.54	42.79	29.34	**29.15**	31.81	18.21	**49.17**	20.56	29.71
G-DeltaNet	45.46	26.47	46.04	23.55	37.28	29.59	66.05	50.75	42.21	20.53	23.34	28.55	14.98	44.91	16.48	24.80
Comba-iplr	**35.37**	24.31	48.15	23.04	38.01	**31.71**	65.83	51.62	43.06	27.98	27.66	28.92	17.96	47.75	18.35	28.10
Comba-splr	39.91	**24.15**	**48.56**	24.32	38.18	30.98	**66.73**	51.41	**43.36**	38.51	27.61	30.07	16.38	48.60	**21.22**	30.40
w/o. $\langle q, dk\rangle$	44.91	25.49	47.94	22.78	37.93	28.96	66.43	50.67	42.45	26.33	28.02	30.03	15.64	48.93	20.36	28.21
w/o. $\tilde{\beta} = b \odot \beta$	40.17	24.56	48.37	24.36	38.02	31.18	65.53	51.36	43.13	35.66	27.70	29.31	16.72	48.10	20.84	29.72
w/o. $\alpha \sim 1$	42.05	25.11	48.33	22.94	37.28	30.44	66.38	50.75	42.68	34.31	25.60	29.44	16.24	46.89	19.74	28.70
w/o. output gate	40.16	24.71	48.29	23.16	37.32	29.48	66.70	50.86	42.64	31.79	23.62	29.63	18.53	48.52	20.84	28.82
w. Initial d=1	39.37	24.27	47.98	22.87	**38.36**	30.66	66.65	**53.04**	43.26	30.24	26.24	28.82	15.68	48.04	18.64	27.94
w. mamba-like	43.20	25.45	46.04	22.95	36.96	30.76	64.36	48.78	41.64	30.64	25.31	28.44	17.33	46.68	18.82	27.87
1.3B params with 100B training tokens and 1M batchsize tokens																
Trans++*	19.29	17.61	55.01	**28.07**	49.21	40.95	70.08	**56.27**	49.93	**44.32**	32.43	**42.59**	24.49	58.47	21.56	**37.31**
RetNet*	21.97	18.18	57.49	26.88	48.09	37.75	69.37	53.28	48.81	13.62	22.59	33.46	15.43	53.79	19.79	26.45
GLA*	19.66	17.61	55.18	27.56	48.89	40.03	69.86	53.91	49.24	27.61	30.93	35.04	22.27	56.28	19.45	31.93
Mamba*	19.01	17.12	56.22	28.01	50.01	42.05	70.36	54.49	50.19	13.90	25.40	33.20	18.50	53.50	21.70	27.70
G-DeltaNet	18.80	17.14	56.82	27.39	49.77	39.94	71.76	51.78	49.58	30.25	27.65	34.06	23.22	58.23	20.36	32.29
Comba-iplr	13.58	16.51	57.11	27.99	51.34	44.40	71.16	52.64	50.77	32.06	28.96	34.83	22.08	57.03	21.03	32.67
Comba-splr	13.39	16.19	**58.54**	27.90	52.64	44.21	**72.03**	55.33	51.78	41.69	35.33	36.14	23.69	58.53	**22.85**	36.37
w. Initial d=1	**12.68**	**16.01**	58.42	27.73	**53.02**	**44.94**	71.76	55.56	**51.91**	42.14	**38.24**	35.47	**25.28**	**59.30**	21.31	36.96
w/o. $\langle q, k\rangle$	15.64	16.94	55.39	26.02	50.30	44.65	68.82	53.12	49.72	36.97	33.55	33.96	23.66	58.12	20.65	34.49

†. For certain reasons, we conduct our evaluation on the lambda-standard dataset rather than on lambda-openai as used in models such as Gated-Deltanet. Consequently, the PPL metric may not be directly comparable with those reported in prior work.

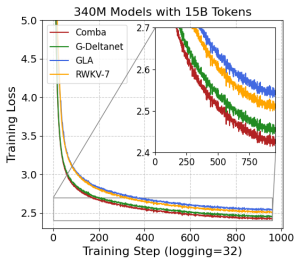

Figure 4: Training loss on 8× A800 GPUs with logging 32.

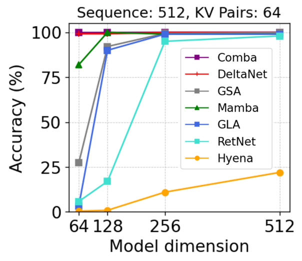

Figure 5: Results on synthetic MQAR task with settings in [2].

Table 7: Recall-intensive tasks in [3] for hybrid architectures, where the quadratic attention component is implemented using `FlashAttention` [25]. The input length is truncated to 2K.

	FDA	SWDE	SQD.	NQ	TQA.	Drop	Avg.
340M params with 15B tokens							
Trans++	46.14	25.87	**33.22**	18.94	45.97	19.94	31.68
GLA-H	58.76	39.46	31.47	16.98	43.80	17.68	34.69
GSA-H	**62.13**	**45.36**	31.17	**20.62**	43.78	18.78	**36.97**
RWKV7-H	43.60	32.71	33.15	18.02	**50.00**	19.74	32.87
GDN-H	52.13	38.80	29.90	15.20	35.55	17.73	31.55
Comba-H	60.47	37.84	31.70	16.48	47.15	**20.11**	35.63

KV cache. However, Comba exhibits recall performance very close to that of transformers. (iii) Ablation studies reveal that output correction significantly contributes to Comba's recall performance, optimizing the similarity of $\boldsymbol{qk}$ will enhance the model's ability to retrieve precise memories, thereby improving recall performance. (iv) In Fig. 5, we test Comba's synthetic recall ability on the MQAR task. The Bilinear RNNs, both Gated-DeltaNet and Comba, demonstrate perfect recall ability.

Inspired by previous hybrid architectures [105, 89, 14, 28, 60, 19], we empirically replace the second-to-last layer in every eight layers of Comba with softmax attention to boost the model's recall ability. In Table 7, (i) we find that nearly all hybrid architectures outperform transformers even with a few quadratic layers. (ii) The best overall performance comes from the GSA-H [112], as GSA itself is a type of intra-layer hybrid architecture. Computationally, it can be expressed as two GLA operations followed by a softmax operation to address attention sparsity issues [92], thus improving recall ability. In the future, we plan to combine GSA with Comba for a flexible hybrid.

Table 8: Performance on `LongBench` [4] tasks with 10K length based on `lm-evaluation-harness` [33].

	Single QA			*Multi QA*			*Summarization*			*Few-shot*			*Code*		
	NQA.	QQA.	MQA.	HQA.	2WM.	MSQ.	GvR.	QMS.	MNs.	TRE.	TQA.	SAM.	LCC.	RBP.	AVG.
Transformer++	17.03	15.41	11.96	14.22	11.65	10.06	10.04	7.40	**15.14**	0.94	9.14	2.40	15.48	9.20	10.72
RetNet	25.34	24.36	21.30	24.16	23.50	21.22	**14.03**	**11.49**	12.15	7.00	22.94	6.75	17.34	19.32	17.92
GLA	26.80	25.55	23.33	24.23	26.52	23.26	11.94	7.19	9.03	**9.54**	27.63	5.30	**20.80**	**21.32**	18.75
Gated-DeltaNet	27.62	27.55	23.34	25.19	26.63	20.26	12.33	7.24	10.51	6.83	28.42	6.07	20.37	18.12	18.61
Comba-splr	**27.73**	**28.56**	**25.78**	**27.49**	**29.55**	**23.34**	11.61	6.20	9.47	6.58	**29.63**	**7.11**	18.04	17.07	**19.16**
w/o. $\alpha \sim 1$	27.53	27.12	24.55	27.30	28.73	23.12	12.42	7.01	9.04	6.86	28.14	6.88	17.65	17.49	18.85

Table 9: Performance on the ImageNet-1K [27] classification, compared to Vision Mamba [115] (linear), DeiT [94] (quadratic), and Agent Attention [44] (sparse).

Model	Res.	Params.	FLOPs	Top-1
DeiT-T	224*224	5.7(MB)	1.2(G)	72.2%
Agent-T	224*224	6.0	1.2	74.9(+2.7)
Vim-T	224*224	7.0	1.5	76.1(+3.9)
Comba-T	**224*224**	**5.8**	**1.1**	**76.3(+4.1)**
DeiT-S	224*224	22.1	4.6	79.8
Agent-S	224*224	22.7	4.6	80.5(+0.7)
Vim-S	224*224	26.0	5.1	80.3(+0.5)
Comba-S	**224*224**	**22.6**	**4.4**	**80.5(+0.7)**

Table 10: Performance on the object tracking datasets such as GOT10k [54] and LaSOT [30], compared to baselines including Vision Mamba (linear), Agent Attention (sparse), and Mixformer [104] (quadratic).

Model	**GOT10k**			**LaSOT**		
	AO	$SR_{0.5}$	$SR_{0.75}$	Suc.	N-Pre.	Pre.
SA	0.704	0.796	0.675	0.690	0.785	0.749
Agent-A	0.695	0.787	0.662	0.644	0.731	0.689
mamba vision	0.700	0.789	0.673	0.677	0.771	0.730
Comba-splr	0.715	0.804	0.686	0.693	**0.789**	0.751
Comba-iplr	**0.718**	**0.809**	**0.688**	**0.694**	0.786	**0.755**

Long-context Modeling Ability As shown in Table 8, Comba generally outperforms the other architectures, with a significant lead in QA and Few-shot tasks. However, it lags behind Gated-DeltaNet in summarization and code tasks, and the underlying reasons warrant further exploration in the future. Additionally, when the special initialization method in §4 is removed, the model's long-sequence modeling capability decreases, which supports the effectiveness of our approach.

4.3 Vision Modeling

Classification As shown in Table 9, Comba achieves SOTA efficiency-accuracy trade-offs across all model scales. For tiny variants, Comba-T improves Top-1 accuracy by 4.1% over DeiT-T with similar parameter count and 8.3% fewer FLOPs. Notably, Comba-T outperforms both Agent-T and Vim-T in accuracy despite requiring fewer computational resources. At small scales, Comba-S matches Agent-S in accuracy while reducing FLOPs by 4.3% and using fewer parameters than Vim-S.

Object tracking To further validate Comba's cross-domain capabilities, we extend experiments to object tracking tasks on GOT-10k and LaSOT datasets. Unlike static image classification, tracking demands efficient temporal modeling and robustness to appearance variations. As shown in Table 10, Comba variants consistently outperform standard attention mechanisms, with the highest AO (0.718) and $SR_{0.75}$ (0.688), exceeding Softmax Attention by 1.4% and 1.3%. Besides, Comba (splr) closely matches Softmax Attention without added computational cost. These results underscore Comba's ability to capture long-range dependencies, such as occlusion recovery and motion continuity.

5 Conclusion & Future Work

This paper provides a comprehensive summary of the development of recursive models in efficient sequence modeling methods and highlights the reasons behind the success of the latest generation of Bilinear RNNs. Drawing on closed-loop control theory, we propose Comba, a new architecture that incorporates both state feedback and output correction, based on SPLR state transformations. We also implement a chunk-wise parallel operator using Triton. Extensive experimental results demonstrate the practical advantages of Comba. However, this paper also has several limitations. For instance, due to limited computational resources, the experimental scale was not extended to larger models, such as the 2.7B model (which typically requires *32×120 GPU hours*). Additionally, since models like Titans, Lattice, and MIRAS have not yet been open-sourced, direct comparisons with these models are difficult. In the future, we will focus on addressing the chunk-wise parallel optimization of these models and explore the integration of GSA with the Comba architecture in an elegant hybrid.

Acknowledgments and Disclosure of Funding

Here, I would like to express my gratitude to those who helped me when I first entered this field, as well as the flash-linear-attention community for their valuable feedback on this paper.

This work is mainly supported by the Guangdong Basic and Applied Basic Research Foundation (No. 2025A1515011994). This work is also supported by the National Natural Science Foundation of China (No. 62402414), Tencent (CCF-Tencent Open Fund, Tencent Rhino-Bird Focused Research Program), Didi (CCF-DiDi GAIA Collaborative Research Funds), Guangzhou Municipal Science and Technology Project (No. 2023A03J0011), Huawei Industrial Funds, and the Guangzhou Industrial Information and Intelligent Key Laboratory Project (No. 2024A03J0628).

References

[1] Ido Amos, Jonathan Berant, and Ankit Gupta. Never train from scratch: Fair comparison of long-sequence models requires data-driven priors. arXiv preprint arXiv:2310.02980, 2023.

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

[3] Simran Arora, Aman Timalsina, Aaryan Singhal, Benjamin Spector, Sabri Eyuboglu, Xinyi Zhao, Ashish Rao, Atri Rudra, and Christopher Ré. Just read twice: closing the recall gap for recurrent language models. arXiv preprint arXiv:2407.05483, 2024.

[4] Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 3119–3137, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[5] 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. arXiv preprint arXiv:2405.04517, 2024.

[6] Ali Behrouz, Meisam Razaviyayn, Peilin Zhong, and Vahab Mirrokni. It's all connected: A journey through test-time memorization, attentional bias, retention, and online optimization. arXiv preprint arXiv:2504.13173, 2025.

[7] Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. arXiv preprint arXiv:2501.00663, 2024.

[8] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. arXiv preprint arXiv:2004.05150, 2020.

[9] Assaf Ben-Kish, Itamar Zimerman, Shady Abu-Hussein, Nadav Cohen, Amir Globerson, Lior Wolf, and Raja Giryes. Decimamba: Exploring the length extrapolation potential of mamba. arXiv preprint arXiv:2406.14528, 2024.

[10] Jacob Benesty, Constantin Paleologu, Tomas Gänsler, and Silviu Ciochină. Recursive least-squares algorithms. In A perspective on stereophonic acoustic echo cancellation, pages 63–69. Springer, 2011.

[11] Aviv Bick, Tobias Katsch, Nimit Sohoni, Arjun Desai, and Albert Gu. Llamba: Scaling distilled recurrent models for efficient language processing. arXiv preprint arXiv:2502.14458, 2025.

[12] Aviv Bick, Kevin Li, Eric Xing, J Zico Kolter, and Albert Gu. Transformers to ssms: Distilling quadratic knowledge to subquadratic models. Advances in Neural Information Processing Systems, 37:31788–31812, 2024.

[13] Christian Bischof and Charles Van Loan. The wy representation for products of householder matrices. SIAM Journal on Scientific and Statistical Computing, 8(1):s2–s13, 1987.

[14] Aaron Blakeman, Aarti Basant, Abhinav Khattar, Adithya Renduchintala, Akhiad Bercovich, Aleksander Ficek, Alexis Bjorlin, Ali Taghibakhshi, Amala Sanjay Deshmukh, Ameya Sunil Mahabaleshwarkar, et al. Nemotron-h: A family of accurate and efficient hybrid mamba-transformer models. arXiv preprint arXiv:2504.03624, 2025.

[15] Guy E Blelloch. Prefix sums and their applications. 1990.

[16] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers, pages 177–186. Springer, 2010.

[17] Carlo Bruni, Gianni DiPillo, and Giorgio Koch. Bilinear systems: An appealing class of" nearly linear" systems in theory and applications. IEEE Transactions on automatic control, 19(4):334–348, 1974.

[18] Snehashish Chakraverty, Deepti Moyi Sahoo, Nisha Rani Mahato, Snehashish Chakraverty, Deepti Moyi Sahoo, and Nisha Rani Mahato. Hebbian learning rule. Concepts of Soft Computing: Fuzzy and ANN with Programming, pages 175–182, 2019.

[19] Xiuwei Chen, Sihao Lin, Xiao Dong, Zisheng Chen, Meng Cao, Jianhua Han, Hang Xu, and Xiaodan Liang. Transmamba: Fast universal architecture adaption from transformers to mamba. arXiv preprint arXiv:2502.15130, 2025.

[20] Yingfa Chen, Xinrong Zhang, Shengding Hu, Xu Han, Zhiyuan Liu, and Maosong Sun. Stuffed mamba: State collapse and state capacity of rnn-based long-context modeling. arXiv preprint arXiv:2410.07145, 2024.

[21] 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. arXiv preprint arXiv:1406.1078, 2014.

[22] 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.

[23] James J Collins and Carlo J De Luca. Open-loop and closed-loop control of posture: a random-walk analysis of center-of-pressure trajectories. Experimental brain research, 95:308–318, 1993.

[24] JJ Collins, CJ De Luca, A Burrows, and LA Lipsitz. Age-related changes in open-loop and closed-loop postural control mechanisms. Experimental brain research, 104:480–492, 1995.

[25] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. Advances in neural information processing systems, 35:16344–16359, 2022.

[26] Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. arXiv preprint arXiv:2405.21060, 2024.

[27] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition, pages 248–255. Ieee, 2009.

[28] Xin Dong, Yonggan Fu, Shizhe Diao, Wonmin Byeon, Zijia Chen, Ameya Sunil Mahabaleshwarkar, Shih-Yang Liu, Matthijs Van Keirsbilck, Min-Hung Chen, Yoshi Suhara, et al. Hymba: A hybrid-head architecture for small language models. arXiv preprint arXiv:2411.13676, 2024.

[29] Jusen Du, Weigao Sun, Disen Lan, Jiaxi Hu, and Yu Cheng. Mom: Linear sequence modeling with mixture-of-memories. arXiv preprint arXiv:2502.13685, 2025.

[30] Lin L. T. Fan H., Bai H. X. and et al. Lasot: A high-quality large-scale single object tracking benchmark. International Journal of Computer Vision, 129:439—-461, 2021.

[31] Nabil H Farhat, Demetri Psaltis, Aluizio Prata, and Eung Paek. Optical implementation of the hopfield model. Applied optics, 24(10):1469–1475, 1985.

[32] 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. arXiv preprint arXiv:2212.14052, 2022.

[33] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 07 2024.

[34] Samuel J Gershman, Ila Fiete, and Kazuki Irie. Key-value memory in the brain. arXiv preprint arXiv:2501.02950, 2025.

[35] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. arXiv preprint arXiv:2012.14913, 2020.

[36] Riccardo Grazzi, Julien Siems, Arber Zela, Jörg KH Franke, Frank Hutter, and Massimiliano Pontil. Unlocking state-tracking in linear rnns through negative eigenvalues. arXiv preprint arXiv:2411.12537, 2024.

[37] Joseph F Grcar. Mathematicians of gaussian elimination. Notices of the AMS, 58(6):782–792, 2011.

[38] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. arXiv preprint arXiv:2312.00752, 2023.

[39] Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. Hippo: Recurrent memory with optimal polynomial projections. Advances in neural information processing systems, 33:1474–1487, 2020.

[40] 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, 2022.

[41] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. arXiv preprint arXiv:2111.00396, 2021.

[42] Albert Gu, Isys Johnson, Aman Timalsina, Atri Rudra, and Christopher Ré. How to train your hippo: State space models with generalized orthogonal basis projections. arXiv preprint arXiv:2206.12037, 2022.

[43] Ankit Gupta, Albert Gu, and Jonathan Berant. Diagonal state spaces are as effective as structured state spaces. Advances in Neural Information Processing Systems, 35:22982–22994, 2022.

[44] Dongchen Han, Tianzhu Ye, Yizeng Han, Zhuofan Xia, Siyuan Pan, Pengfei Wan, Shiji Song, and Gao Huang. Agent attention: On the integration of softmax and linear attention. In Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part L, page 124–140. Springer, 2024.

[45] Håkan Hjalmarsson. From experiment design to closed-loop control. Automatica, 41(3):393–438, 2005.

[46] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

[47] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. ICLR, 1(2):3, 2022.

[48] Jiaxi Hu, Disen Lan, Ziyu Zhou, Qingsong Wen, and Yuxuan Liang. Time-ssm: Simplifying and unifying state space models for time series forecasting. arXiv preprint arXiv:2405.16312, 2024.

[49] Zheyuan Hu, Nazanin Ahmadi Daryakenari, Qianli Shen, Kenji Kawaguchi, and George Em Karniadakis. State-space models are accurate and efficient neural operators for dynamical systems. arXiv preprint arXiv:2409.03231, 2024.

[50] Kazuki Irie, Imanol Schlag, Róbert Csordás, and Jürgen Schmidhuber. Going beyond linear transformers with recurrent fast weight programmers. Advances in neural information processing systems, 34:7703–7717, 2021.

[51] Thierry Joffrain, Tze Meng Low, Enrique S Quintana-Ortí, Robert van de Geijn, and Field G Van Zee. Accumulating householder transformations, revisited. ACM Transactions on Mathematical Software (TOMS), 32(2):169–179, 2006.

[52] Mahdi Karami and Vahab Mirrokni. Lattice: Learning to efficiently compress the memory. arXiv preprint arXiv:2504.05646, 2025.

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

[54] X. Zhao L. Huang and K. Huang. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. IEEE Transactions on Pattern Analysis and Machine Intelligence, 43(5):1562–1577, 2021.

[55] Disen Lan, Weigao Sun, Jiaxi Hu, Jusen Du, and Yu Cheng. Liger: Linearizing large language models to gated recurrent structures. arXiv preprint arXiv:2503.01496, 2025.

[56] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, Fujie Huang, et al. A tutorial on energy-based learning. Predicting structured data, 1(0), 2006.

[57] Steven J Leon, Åke Björck, and Walter Gander. Gram-schmidt orthogonalization: 100 years and more. Numerical Linear Algebra with Applications, 20(3):492–532, 2013.

[58] Aonian Li, Bangwei Gong, Bo Yang, Boji Shan, Chang Liu, Cheng Zhu, Chunhao Zhang, Congchao Guo, Da Chen, Dong Li, et al. Minimax-01: Scaling foundation models with lightning attention. arXiv preprint arXiv:2501.08313, 2025.

[59] Haoyang Li, Yiming Li, Anxin Tian, Tianhao Tang, Zhanchao Xu, Xuejia Chen, Nicole Hu, Wei Dong, Qing Li, and Lei Chen. A survey on large language model acceleration based on kv cache management. arXiv preprint arXiv:2412.19442, 2024.

[60] Yixing Li, Ruobing Xie, Zhen Yang, Xingwu Sun, Shuaipeng Li, Weidong Han, Zhanhui Kang, Yu Cheng, Chengzhong Xu, Di Wang, et al. Transmamba: Flexibly switching between transformer and mamba. arXiv preprint arXiv:2503.24067, 2025.

[61] Zhixuan Lin, Evgenii Nikishin, Xu Owen He, and Aaron Courville. Forgetting transformer: Softmax attention with a forget gate. arXiv preprint arXiv:2503.02130, 2025.

[62] Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, Jianbin Jiao, and Yunfan Liu. Vmamba: Visual state space model. Advances in neural information processing systems, 37:103031–103063, 2024.

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

[64] Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, et al. Moba: Mixture of block attention for long-context llms. arXiv preprint arXiv:2502.13189, 2025.

[65] ROBERTJ McEliece, Edwardc Posner, EUGENER Rodemich, and SANTOSHS Venkatesh. The capacity of the hopfield associative memory. IEEE transactions on Information Theory, 33(4):461–482, 1987.

[66] Larry R Medsker, Lakhmi Jain, et al. Recurrent neural networks. Design and Applications, 5(64-67):2, 2001.

[67] Tsendsuren Munkhdalai, Alessandro Sordoni, Tong Wang, and Adam Trischler. Metalearned neural memory. Advances in Neural Information Processing Systems, 32, 2019.

[68] Tsendsuren Munkhdalai and Hong Yu. Neural semantic encoders. In Proceedings of the conference. Association for Computational Linguistics. Meeting, volume 1, page 397, 2017.

[69] Daniele Paliotta, Junxiong Wang, Matteo Pagliardini, Kevin Y Li, Aviv Bick, J Zico Kolter, Albert Gu, François Fleuret, and Tri Dao. Thinking slow, fast: Scaling inference compute with distilled reasoners. arXiv preprint arXiv:2502.20339, 2025.

[70] Panos M Pardalos and Vitaliy A Yatsenko. Optimization and control of bilinear systems: theory, algorithms, and applications, volume 11. Springer Science & Business Media, 2010.

[71] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[72] Badri Narayana Patro and Vijay Srinivas Agneeswaran. Mamba-360: Survey of state space models as transformer alternative for long sequence modelling: Methods, applications, and challenges. arXiv preprint arXiv:2404.16112, 2024.

[73] Bo Peng, Ruichong Zhang, Daniel Goldstein, Eric Alcaide, Haowen Hou, Janna Lu, William Merrill, Guangyu Song, Kaifeng Tan, Saiteja Utpala, et al. Rwkv-7" goose" with expressive dynamic state evolution. arXiv preprint arXiv:2503.14456, 2025.

[74] George Philipp, Dawn Song, and Jaime G Carbonell. The exploding gradient problem demystified-definition, prevalence, impact, origin, tradeoffs, and solutions. arXiv preprint arXiv:1712.05577, 2017.

[75] DL Prados and SC Kak. Neural network capacity using delta rule. Electronics Letters, 25(3):197–199, 1989.

[76] Ofir Press, Noah A Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. arXiv preprint arXiv:2108.12409, 2021.

[77] Zhen Qin, Weigao Sun, Dong Li, Xuyang Shen, Weixuan Sun, and Yiran Zhong. Lightning attention-2: A free lunch for handling unlimited sequence lengths in large language models. arXiv preprint arXiv:2401.04658, 2024.

[78] Zhen Qin, Weigao Sun, Dong Li, Xuyang Shen, Weixuan Sun, and Yiran Zhong. Various lengths, constant speed: Efficient language modeling with lightning attention. arXiv preprint arXiv:2405.17381, 2024.

[79] Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. Hgrn2: Gated linear rnns with state expansion. arXiv preprint arXiv:2404.07904, 2024.

[80] 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. arXiv preprint arXiv:2406.07522, 2024.

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

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

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

[84] Jimmy TH Smith, Andrew Warrington, and Scott W Linderman. Simplified state space layers for sequence modeling. arXiv preprint arXiv:2208.04933, 2022.

[85] 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.net/blog/slimpajama-a-627b-token-cleaned-and-deduplicated-version-of-redpajama, 2023.

[86] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. Neurocomputing, 568:127063, 2024.

[87] Jingrui Sun, Xun Li, and Jiongmin Yong. Open-loop and closed-loop solvabilities for stochastic linear quadratic optimal control problems. SIAM Journal on Control and Optimization, 54(5):2274–2308, 2016.

[88] Weigao Sun, Disen Lan, Tong Zhu, Xiaoye Qu, and Yu Cheng. Linear-moe: Linear sequence modeling meets mixture-of-experts. arXiv preprint arXiv:2503.05447, 2025.

[89] Xingwu Sun, Yanfeng Chen, Yiqing Huang, Ruobing Xie, Jiaqi Zhu, Kai Zhang, Shuaipeng Li, Zhen Yang, Jonny Han, Xiaobo Shu, et al. Hunyuan-large: An open-source moe model with 52 billion activated parameters by tencent. arXiv preprint arXiv:2411.02265, 2024.

[90] Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, et al. Learning to (learn at test time): Rnns with expressive hidden states. arXiv preprint arXiv:2407.04620, 2024.

[91] 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.

[92] Shawn Tan, Yikang Shen, Songlin Yang, Aaron Courville, and Rameswar Panda. Stick-breaking attention. arXiv preprint arXiv:2410.17980, 2024.

[93] Philippe Tillet, Hsiang-Tsung Kung, and David Cox. Triton: an intermediate language and compiler for tiled neural network computations. In Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages, pages 10–19, 2019.

[94] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers & distillation through attention. In Proceedings of the 38th International Conference on Machine Learning, pages 10347–10357. PMLR, 2021.

[95] 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.

[96] 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.

[97] 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.

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

[99] Xinyue Wang, Junxia Ma, and Weili Xiong. Expectation-maximization algorithm for bilinear state-space models with time-varying delays under non-gaussian noise. International Journal of Adaptive Control and Signal Processing, 37(10):2706–2724, 2023.

[100] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. Advances in neural information processing systems, 35:24824–24837, 2022.

[101] Bernard Widrow and Marcian E Hoff. Adaptive switching circuits. In Neurocomputing: foundations of research, pages 123–134. 1988.

[102] Darrell Williamson. Observation of bilinear systems with application to biological control. Automatica, 13(3):243–254, 1977.

[103] Ruyi Xu, Guangxuan Xiao, Haofeng Huang, Junxian Guo, and Song Han. Xattention: Block sparse attention with antidiagonal scoring. arXiv preprint arXiv:2503.16428, 2025.

[104] L. Wang Y. Cui, C. Jiang and G. Wu. Mixformer: End-to-end tracking with iterative mixed attention. In 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 13598–13608. CVPR Organization, 2022.

[105] Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. arXiv preprint arXiv:2412.06464, 2024.

[106] 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, 2025.

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

[108] Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. arXiv preprint arXiv:2406.06484, 2024.

[109] Songlin Yang and Yu Zhang. Fla: A triton-based library for hardware-efficient implementations of linear attention mechanism, January 2024.

[110] Zhifan Ye, Kejing Xia, Yonggan Fu, Xin Dong, Jihoon Hong, Xiangchi Yuan, Shizhe Diao, Jan Kautz, Pavlo Molchanov, and Yingyan Celine Lin. Longmamba: Enhancing mamba's long context capabilities via training-free receptive field enlargement. arXiv preprint arXiv:2504.16053, 2025.

[111] Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, YX Wei, Lean Wang, Zhiping Xiao, et al. Native sparse attention: Hardware-aligned and natively trainable sparse attention. arXiv preprint arXiv:2502.11089, 2025.

[112] 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.

[113] Yingbo Zhao and Jorge Cortés. Gramian-based reachability metrics for bilinear networks. IEEE Transactions on Control of Network Systems, 4(3):620–631, 2016.

[114] Shu Zhong, Mingyu Xu, Tenglong Ao, and Guang Shi. Understanding transformer from the perspective of associative memory. arXiv preprint arXiv:2505.19488, 2025.

[115] Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision mamba: efficient visual representation learning with bidirectional state space model. In Proceedings of the 41st International Conference on Machine Learning, pages 62429–62442. JMLR.org, 2024.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: As shown in the abstract and the end of the introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discussed the limitations in the last section.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We provide the proof/derivation in the Appendix.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We state the experiment setting before the results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide the code in an anonymous link.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We state the experiment setting before the results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We conducted the evaluation using open-source tools that inherently include statistical significance, though we chose not to present these results in the paper.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provided the experimental details and operator speed.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We follow the Code of ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We state our work has the potential to become a foundation structure for Nonlinear RNN.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [No]

    Justification: We use open-source datasets.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have cited corresponding works in the paper.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Our code in the anonymous link has a detailed readme.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: the paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: the paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: the core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.

Supplementary Material

Improving Bilinear RNNs with Closed-loop Control

Table of Contents

A Additional Background **25**
A.1 Bilinear Systems . . . 25
A.2 Comba in a State Space Model Perspective . . . 25

B Operator Implementation Derivation **25**
B.1 Recurrent Implementation for Comba . . . 25
B.2 WY Representation & UT Transform . . . 26
B.3 Comba-SPLR-pk . . . 26

A Additional Background

A.1 Bilinear Systems

Formally, TTT, Gated-DeltaNet, RWKV-7, and Comba should be classified as bilinear systems [17, 113, 99, 102, 70]. These systems are linear with respect to both $\boldsymbol{S}$ and $\boldsymbol{K}, \boldsymbol{V}$, but due to the interaction between $\boldsymbol{S}$ and $\boldsymbol{K}$, the overall system is nonlinear. Typically considered a special type of nonlinear system, bilinear systems possess more expressive power than linear systems while remaining more controllable than strictly nonlinear systems, such as chaotic systems. They are widely applied in the study of physical systems and biological population dynamics [102, 70].

A.2 Comba in a State Space Model Perspective

Table 11: Update rules in a control & neural-memory perspective, with feedback $\boldsymbol{P}(\cdot)$ and scalar factor d.

Option	Open-loop Control (Mamba2)	Close-loop Control (Comba)	Gated Delta Rule
Input / Memorize	$\boldsymbol{x}_t = \exp(\Delta_t A)\boldsymbol{x}_{t-1} + \Delta_t \boldsymbol{B}_t \boldsymbol{u}_t$	$\boldsymbol{x}_t = \exp(\Delta_t A)\boldsymbol{x}_{t-1} + \Delta_t \boldsymbol{B}_t \boldsymbol{u}_t^{\text{new}}$	$\boldsymbol{S}_t = \alpha_t \boldsymbol{S}_{t-1} + \beta_t \boldsymbol{v}_t^{\text{new}} \boldsymbol{k}_t^{\intercal}$
Feedback / Reflect	nan	$\boldsymbol{u}_t^{\text{new}} = \boldsymbol{u}_t - \boldsymbol{P}_t(\boldsymbol{x}_{t-1})$	$\boldsymbol{v}_t^{\text{new}} = \boldsymbol{v}_t - \alpha_t \boldsymbol{S}_{t-1}\boldsymbol{k}_t$
Output / Recollect	$\boldsymbol{o}_t = \boldsymbol{C}_t \boldsymbol{x}_t + \boldsymbol{D}\boldsymbol{u}_t$	$\boldsymbol{o}_t = \boldsymbol{C}_t \boldsymbol{x}_t + \boldsymbol{D}(\boldsymbol{u}_t - \boldsymbol{P}_t(\boldsymbol{x}_t))$	$\boldsymbol{o}_t = \boldsymbol{S}_t \boldsymbol{q}_t$

In Table 11, we provide an explanation of Comba from the perspective of state-space models. Here, β_t corresponds to Δ_t, which is an Euler discretization. In state-space models, there is typically a residual term $D\boldsymbol{u}_t$, which we integrate into the residual connection in our framework to omit it.

B Operator Implementation Derivation

B.1 Recurrent Implementation for Comba

We provide the recurrent Comba in Algorithm 1.

```
def Recurrent_comba(q, k, v, alpha, beta, b, d):
    B, T, H, D = q.shape
    q_new = q - d * k # Output correction
    o, S = torch.zeros_like(v), torch.zeros(b, h, d, d)
    for i in range (T):
        _q, _k, _alpha, _beta = q_new[:, i], k[:, i], alpha[:, i],
    beta[:, i]
```

```
        _v_new = _beta[..., None] * (v[:, i] - b * (S * _k[..., None])
    .sum(-2))
        S = _At[..., None] * S + _k.unsqueeze(-1) * _v_new.unsqueeze
    (-2)
        o[:, i] = torch.einsum('bhd,bhdm->bhm', _q, S)
    return o
```

Listing 1: Recurrent Comba-pk in Pytorch-like Pseudo-code for Inference

B.2 WY Representation & UT Transform

Detailed derivation can be found in the Appendix of DeltaNet [108, 105].

B.3 Comba-SPLR-pk

We also provide an alternative implementation, where $b\boldsymbol{k}$ is integrated into $\boldsymbol{p}$ to make it more close to control theory, these two are equivalent.

```
def Recurrent_comba(q, k, v, p, At, dt, D):
    b, t, h, d = q.shape
    q_new = q - D[..., None] * p # Output correction
    o, S = torch.zeros_like(v), torch.zeros(b, h, d, d)
    for i in range(t):
        _q, _k, _p, _At, _dt = q_new[:, i], k[:, i], p[:, i], At[:, i
    ], dt[:, i]
        _v_new = _dt[..., None] * (v[:, i] - (S * _p[..., None]).sum
    (-2))
        S = _At[..., None] * S + _k.unsqueeze(-1) * _v_new.unsqueeze
    (-2)
        o[:, i] = torch.einsum('bhd,bhdm->bhm', _q, S)
    return o
```

Listing 2: Recurrent Comba-pk in Pytorch-like Pseudo-code for Inference

By partially expanding the recurrence for Eq. 5, we have

$$\boldsymbol{S}^r_{[t]} = \boldsymbol{S}^0_{[t]} \underbrace{\left(\prod_{i=1}^{r}\left(\alpha^i_{[t]} - \beta^i_{[t]}\boldsymbol{p}^i_{[t]}\boldsymbol{k}^{i\intercal}_{[t]}\right)\right)}_{:=\boldsymbol{D}^r_{[t]}\ \text{(“pseudo” memory decay)}} + \underbrace{\sum_{i=1}^{r}\left(\beta^i_{[t]}\boldsymbol{v}^i_{[t]}\boldsymbol{k}^{i\intercal}_{[t]}\prod_{j=i+1}^{r}\left(\alpha^j_{[t]} - \beta^j_{[t]}\boldsymbol{p}^j_{[t]}\boldsymbol{k}^{j\intercal}_{[t]}\right)\right)}_{:=\boldsymbol{H}^r_{[t]}\ \text{(“pseudo” Incremental memory)}}$$

Then, we employ the WY representation [13]:

$$\boldsymbol{D}^r_{[t]} = \alpha^{1:r}_{[t]} - \sum_{i=1}^{r}\alpha^{i:r}_{[t]}\boldsymbol{w}^i_{[t]}\boldsymbol{k}^{i\intercal}_{[t]} \qquad \boldsymbol{w}^r_{[t]} = \beta^r_{[t]}\left(\alpha^{1:r-1}_{[t]}\boldsymbol{p}^r_{[t]} - \sum_{i=1}^{r-1}\boldsymbol{w}^i_{[t]}\left(\alpha^{i:r-1}_{[t]}\boldsymbol{k}^{i\intercal}_{[t]}\boldsymbol{p}^r_{[t]}\right)\right)$$

$$\boldsymbol{H}^r_{[t]} = \sum_{i=1}^{r}\alpha^{i:r}_{[t]}\boldsymbol{u}^i_{[t]}\boldsymbol{k}^{i\intercal}_{[t]} \qquad \boldsymbol{u}^r_{[t]} = \beta^r_{[t]}\left(\boldsymbol{v}^r_{[t]} - \sum_{i=1}^{r-1}\boldsymbol{u}^i_{[t]}\left(\alpha^{i:r-1}_{[t]}\boldsymbol{k}^{i\intercal}_{[t]}\boldsymbol{p}^r_{[t]}\right)\right)$$

To maximize hardware efficiency, we apply the UT transform [51] to reduce non-matmul FLOPs, which is crucial to enable better hardware utilization during training.

$$\boldsymbol{W}_{[t]} = \boldsymbol{M}_{[t]}\mathrm{Diag}\left(\beta^{1\to C}_{[t]} \odot \alpha^{0\to(C-1)}_{[t]}\right)\boldsymbol{P}_{[t]}, \qquad \boldsymbol{U}_{[t]} = \boldsymbol{M}_{[t]}\mathrm{Diag}\left(\beta^{1\to C}_{[t]}\right)\boldsymbol{V}_{[t]}$$

$$\boldsymbol{M}_{[t]} = \left(\boldsymbol{I} + \mathrm{lower}\left(\mathrm{Diag}\left(\beta^{1\to C}_{[t]}\right)\left(\mathcal{A}^{(i-1)/j}_{[t]} \odot \boldsymbol{P}_{[t]}\boldsymbol{K}^{\intercal}_{[t]}\right)\right)\right)^{-1}$$

The inverse of a lower triangular matrix can be efficiently computed through an iterative row-wise approach by forward substitution in Gaussian elimination [37] and maintain data in float32.

Then we have the following vector form:

$$\boldsymbol{S}_{[t]}^r = \boldsymbol{S}_{[t]}^0 \boldsymbol{D}_{[t]}^r = \alpha_{[t]}^{1:r} \boldsymbol{S}_{[t]}^0 + \sum_{i=1}^{r} \alpha_{[t]}^{i:r} \left(\boldsymbol{u}_{[t]}^r - \left(\boldsymbol{S}_{[t]}^0 \boldsymbol{w}_{[t]}^i \right) \right) \boldsymbol{k}_{[t]}^{i\intercal}$$

$$\boldsymbol{o}_{[t]}^r = \boldsymbol{S}_{[t]}^r \tilde{\boldsymbol{q}}_{[t]}^r = \alpha_{[t]}^{1:r} \boldsymbol{S}_{[t]}^0 \tilde{\boldsymbol{q}}_{[t]}^r + \sum_{i=1}^{r} \left(\boldsymbol{u}_{[t]}^r - \left(\boldsymbol{S}_{[t]}^0 \boldsymbol{w}_{[t]}^i \right) \right) \left(\alpha_{[t]}^{i:r} \boldsymbol{k}_{[t]}^{i\intercal} \tilde{\boldsymbol{q}}_{[t]}^r \right)$$

Equivalently, in matrix form:

$$\boldsymbol{S}_{[t+1]} = \alpha_{[t]}^{1:C} \boldsymbol{S}_{[t]} + \left(\boldsymbol{U}_{[t]} - \boldsymbol{W}_{[t]} \boldsymbol{S}_{[t]}^\intercal \right)^\intercal \mathrm{Diag}\left(\alpha_{[t]}^{i \to C} \right) \boldsymbol{K}_{[t]}$$

$$\boldsymbol{O}_{[t]} = \underbrace{\mathrm{Diag}\left(\alpha_{[t]}^{1 \to C} \right) \tilde{\boldsymbol{Q}}_{[t]} \boldsymbol{S}_{[t]}^\intercal}_{\text{inner chunk}} + \underbrace{\mathrm{Tril}(\tilde{\boldsymbol{Q}}_{[t]} \boldsymbol{K}_{[t]}^\intercal \odot \mathcal{A}_{[t]}^{i/j})}_{\text{intra chunk}} \underbrace{\left(\boldsymbol{U}_{[t]} - \boldsymbol{W}_{[t]} \boldsymbol{S}_{[t]}^\intercal \right)}_{\text{``pseudo''-value term}}$$

where the query matrix $\tilde{\boldsymbol{Q}}$ is also influenced by the closed-loop control and can be precomputed by: $\tilde{\boldsymbol{Q}} = \boldsymbol{Q} - \mathrm{Diag}(d_{[t]}^{1 \to C}) \boldsymbol{P}_{[t]}$.