🅚 KIMI LINEAR:
AN EXPRESSIVE, EFFICIENT ATTENTION ARCHITECTURE

Kimi Team
🔗 https://github.com/MoonshotAI/Kimi-Linear

ABSTRACT

We introduce Kimi Linear, a hybrid linear attention architecture that, for the first time, outperforms full attention under fair comparisons across various scenarios—including short-context, long-context, and reinforcement learning (RL) scaling regimes. At its core lies Kimi Delta Attention (KDA), an expressive linear attention module that extends Gated DeltaNet [110] with a finer-grained gating mechanism, enabling more effective use of limited finite-state RNN memory. Our bespoke chunkwise algorithm achieves high hardware efficiency through a specialized variant of the *Diagonal-Plus-Low-Rank* (DPLR) transition matrices, which substantially reduces computation compared to the general DPLR formulation while remaining more consistent with the classical delta rule.

We pretrain a Kimi Linear model with 3B activated parameters and 48B total parameters, based on a layerwise hybrid of KDA and Multi-Head Latent Attention (MLA). Our experiments show that with an identical training recipe, Kimi Linear outperforms full MLA with a sizeable margin across all evaluated tasks, while reducing KV cache usage by up to 75% and achieving up to $6\times$ decoding throughput for a 1M context. These results demonstrate that Kimi Linear can be a drop-in replacement for full attention architectures with superior performance and efficiency, including tasks with longer input and output lengths.

To support further research, we open-source the KDA kernel and vLLM implementations [1], and release the pre-trained and instruction-tuned model checkpoints. [2]
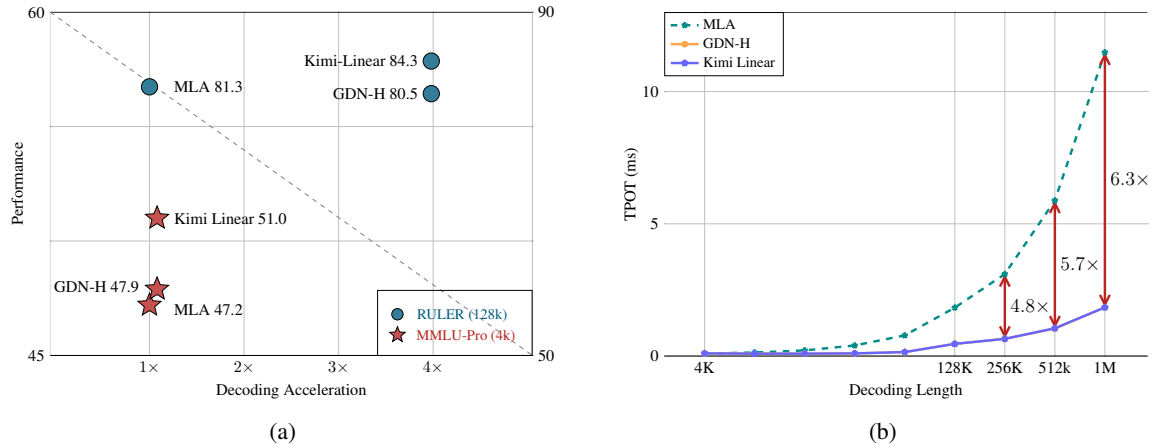

Figure 1: (a) Performance vs. acceleration. With strict fair comparisons with 1.4T training tokens, on MMLU-Pro (4k context length, red stars), Kimi Linear leads performance (51.0) at similar speed. On RULER (128k context length, blue circles), it is Pareto-optimal, achieving top performance (84.3) and $3.98\times$ acceleration. (b) Time per output token (TPOT) vs. decoding length. Kimi Linear (blue line) maintains a low TPOT, matching GDN-H and outperforming MLA at long sequences. This enables larger batches, yielding a $6.3\times$ faster TPOT (1.84ms vs. 11.48ms) than MLA at 1M tokens.

1 Introduction

As large language models (LLMs) evolve into increasingly capable agents [49], the computational demands of inference—particularly in long-horizon and reinforcement learning (RL) settings—are becoming a central bottleneck. This shift toward *RL test-time scaling* [94, 32, 79, 73, 52], where models must process extended trajectories, tool-use interactions, and complex decision spaces at inference time, exposes fundamental inefficiencies in standard attention mechanisms. In particular, the quadratic time complexity and the linearly growing key–value (KV) cache of softmax attention introduce substantial computational and memory overheads, hindering throughput, context-length scaling, and real-time interactivity.

Linear attention [47] offers a principled approach to reducing computational complexity but has historically under-performed softmax attention in language modeling—even for short sequences—due to limited expressivity. Recent advances have significantly narrowed this gap, primarily through two innovations: gating or decay mechanisms [91, 16, 113] and the delta rule [83, 111, 110, 70]. Together, these developments have pushed linear attention closer to softmax-level quality on moderate-length sequences. Nevertheless, purely linear structure remain fundamentally constrained by the finite-state capacity, making long-sequence modeling and in-context retrieval theoretically challenging [103, 4, 44].

Hybrid architectures that combine softmax and linear attention—using a few global-attention layers alongside predominantly faster linear layers—have thus emerged as a practical compromise between quality and efficiency [56, 99, 65, 12, 31, 80]. However, previous hybrid models often operated at limited scale or lacked comprehensive evaluation across diverse benchmarks. The core challenge remains: to develop an attention architecture that matches or surpasses full attention in quality while achieving substantial efficiency gains in both speed and memory—an essential step toward enabling the next generation of agentic, decoding-heavy LLMs.

In this work, we present **Kimi Linear**, a hybrid linear attention architecture designed to meet the efficiency demands of agentic intelligence and test-time scaling without compromising quality. At its core lies **Kimi Delta Attention (KDA)**, a hardware-efficient linear attention module that extends Gated DeltaNet [110] with a finer-grained gating mechanism. While GDN, similar to Mamba2 [16], employs a coarse head-wise forget gate, KDA introduces a channel-wise variant in which each feature dimension maintains an independent forgetting rate, akin to Gated Linear Attention (GLA) [113]. This fine-grained design enables more precise regulation of the finite-state RNN memory, unlocking the potential of RNN-style models within hybrid architectures.

Crucially, KDA parameterizes its transition dynamics with a specialized variant of the *Diagonal-Plus-Low-Rank* (DPLR) matrices, enabling a bespoke chunkwise-parallel algorithm that substantially reduces computation relative to general DPLR formulations while remaining consistent with the classical delta rule.

Kimi Linear interleaves KDA with periodic full attention layers in a uniform 3:1 ratio. This hybrid structure reduces memory and KV-cache usage by up to 75% during long-sequence generation while preserving global information flow via the full attention layers. Through matched-scale pretraining and evaluation, we show that Kimi Linear consistently matches or outperforms strong full-attention baselines across short-context, long-context, and RL-style post-training tasks—while achieving up to $6\times$ higher decoding throughput at 1M context length.

To facilitate further research, we release open-source KDA kernels with vLLM integration, as well as pre-trained and instruction-tuned checkpoints. These components are drop-in compatible with existing full-attention pipelines, requiring no modification to caching or scheduling interfaces, thereby facilitating research on hybrid architectures.

Contributions

- **Kimi Delta Attention (KDA):** a linear attention mechanism that refines the gated delta rule with improved recurrent memory management and hardware efficiency.

- **The Kimi Linear architecture:** a hybrid design adopting a 3:1 KDA-to-global attention ratio, reducing memory footprint while surpassing full-attention quality.

- **Fair empirical validation at scale:** through 1.4T token training runs, Kimi Linear outperforms full attention and other baselines in short/long context and RL-style evaluations, with full release of kernels, vLLM integration, and checkpoints.

2 Preliminary

In this section, we introduce the technical background related to our proposed Kimi Delta Attention.

2.1 Notation

In this paper, we define $\square_t \in \mathbb{R}^{d_k}$ or $\mathbb{R}^{d_v}$, s.t., $\square \in \{\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v}, \boldsymbol{o}, \boldsymbol{u}, \boldsymbol{w}\}$ denotes a t-th query/key/value/output column vectors, and $\mathbf{S}_t \in \mathbb{R}^{d_k \times d_v}$ represents the matrix-form memory state. $\mathbf{M}$ and $\mathbf{M}^-$ denote lower-triangular masks with and without diagonal elements, respectively; for convenience, we also write them as Tril and $\mathrm{StrictTril}$.

Chunk-wise Formulation　Suppose the sequence is split into L/C chunks where each chunk is of length C. We define $\square_{[t]} \in \mathbb{R}^{C \times d}$ for $\square \in \{\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{O}, \mathbf{U}, \mathbf{W}\}$ are matrix that stack the vectors within the t-th chunk, and $\square_{[t]}^r = \square_{tC+r}$ is the r-th element of the chunk. Note that $t \in [0, L/C), r \in [1, C]$. State matrices are also re-indexed such that $\boldsymbol{S}_{[t]}^i = \boldsymbol{S}_{tC+i}$. Additionally, $\boldsymbol{S}_{[t]} := \boldsymbol{S}_{[t]}^0 = \boldsymbol{S}_{[t-1]}^C$, i.e., the initial state of a chunk is the last state of the previous chunk.

Decay Formulation　We define the cumulative decay $\gamma_{[t]}^{i \to j} := \prod_{k=i}^{j} \alpha_{[t]}^k$, and abbreviate $\gamma_{[t]}^{1 \to r}$ as $\gamma_{[t]}^r$. Additionally, $\mathcal{A}_{[t]} := \mathcal{A}_{[t]}^{i/j} \in \mathbb{R}^{C \times C}$ is the matrix with elements $\gamma_{[t]}^i / \gamma_{[t]}^j$. $\mathrm{Diag}\,(\boldsymbol{\alpha}_t)$ denotes the fine-grained decay, $\mathrm{Diag}\left(\gamma_{[t]}^{i \to j}\right) := \prod_{k=i}^{j} \mathrm{Diag}\left(\boldsymbol{\alpha}_{[t]}^k\right)$, and $\boldsymbol{\Gamma}_{[t]}^{i \to j} \in \mathbb{R}^{C \times d_k}$ is the matrix stack from $\gamma_{[t]}^i$ to $\gamma_{[t]}^j$.

2.2 Linear Attention and the Gated Delta Rule

Linear Attention as Online Learning.　Linear attention [47] maintains a matrix-valued recurrent state that accumulates key–value associations:

$$\mathbf{S}_t = \mathbf{S}_{t-1} + \boldsymbol{k}_t \boldsymbol{v}_t^\top, \qquad \boldsymbol{o}_t = \mathbf{S}_t^\top \boldsymbol{q}_t.$$

From the fast-weight perspective [83, 84], $\mathbf{S}_t$ serves as an associative memory storing transient mappings from keys to values. This update can be viewed as performing gradient *ascent* on the unbounded correlation objective

$$\mathcal{L}_t(\mathbf{S}) = \langle \mathbf{S}^\top \boldsymbol{k}_t, \boldsymbol{v}_t \rangle,$$

which continually reinforces recent key–value pairs without any forgetting. However, such an objective provides no criterion for which memories to erase, and the accumulated state grows unbounded, leading to interference over long contexts.

DeltaNet: Online Gradient Descent on Reconstruction Loss.　DeltaNet [83] reinterprets this recurrence as online gradient *descent* on a reconstruction objective:

$$\mathcal{L}_t(\mathbf{S}) = \tfrac{1}{2}\|\mathbf{S}^\top \boldsymbol{k}_t - \boldsymbol{v}_t\|^2.$$

Taking a gradient step with learning rate β_t gives

$$\mathbf{S}_t = \mathbf{S}_{t-1} - \beta_t \nabla_\mathbf{S} \mathcal{L}_t(\mathbf{S}_{t-1}) = (\mathbf{I} - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top)\mathbf{S}_{t-1} + \beta_t \boldsymbol{k}_t \boldsymbol{v}_t^\top.$$

This rule—the classical *delta rule*—treats $\mathbf{S}$ as a learnable associative memory that continually corrects itself toward the mapping $\boldsymbol{k}_t \mapsto \boldsymbol{v}_t$. The rank-1 update structure, equivalent to a generalized Householder transformation, supports hardware-efficient chunkwise parallelization [11, 111].

Gated DeltaNet as Weight Decay.　Although DeltaNet stabilizes learning, it still retains outdated associations indefinitely. Gated DeltaNet (GDN) [110] introduces a scalar forget gate $\alpha_t \in [0, 1]$, yielding

$$\mathbf{S}_t = \alpha_t(\mathbf{I} - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top)\mathbf{S}_{t-1} + \beta_t \boldsymbol{k}_t \boldsymbol{v}_t^\top.$$

Here, α_t acts as a form of *weight decay* on the fast weights [8], implementing a forgetting mechanism analogous to data-dependent L_2 regularization. This simple yet effective modification provides a principled way to control memory lifespan and mitigate interference, improving both stability and long-context generalization while preserving DeltaNet's parallelizable structure.

From this perspective, we observe that GDN can be interpreted as a form of multiplicative positional encoding where the transition matrix is data-dependent and learnable, relaxing the orthogonality constraint of RoPE [114].[3]

---

[3]When the state transformation matrix preserves its orthogonality, absolute positional encodings can also be applied independently to $\boldsymbol{q}$ and $\boldsymbol{k}$ to be converted into relative positional encodings during the attention computation [86].

3 Kimi Delta Attention: Improving Delta Rule with Fine-grained Gating

We propose Kimi Delta Attention (KDA), a new gated linear attention variant that refines GDN's scalar decay by introducing a fine-grained diagonalized gate $\mathrm{Diag}(\boldsymbol{\alpha}_t)$ that enables fine-grained control over memory decay and positional awareness (as discussed in §6.1). We begin by introducing the chunkwise parallelization of KDA, showing how a series of rank-1 matrix transformations can be compressed into a dense representation while maintaining stability under diagonal gating. We then highlight the efficiency gains of KDA over the standard DPLR formulation.

$$\mathbf{S}_t = \left(\mathbf{I} - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top\right) \mathrm{Diag}\left(\boldsymbol{\alpha}_t\right) \mathbf{S}_{t-1} + \beta_t \boldsymbol{k}_t \boldsymbol{v}_t^\top \in \mathbf{R}^{d_k \times d_v}; \qquad \boldsymbol{o}_t = \mathbf{S}_t^\top \boldsymbol{q}_t \in \mathbf{R}^{d_v} \tag{1}$$

3.1 Hardware-Efficient Chunkwise Algorithm

By partially expanding the recurrence for Eq. 1 into a chunk-wise formulation, we have:

$$\mathbf{S}_{[t]}^r = \left(\prod_{i=1}^{r} \left(\mathbf{I} - \beta_{[t]}^i \boldsymbol{k}_{[t]}^i \boldsymbol{k}_{[t]}^{i\top}\right) \mathrm{Diag}(\boldsymbol{\alpha}_{[t]}^i)\right) \cdot \mathbf{S}_{[t]}^0 + \beta_{[t]}^i \boldsymbol{k}_{[t]}^i \boldsymbol{v}_{[t]}^{i\top} \tag{2}$$

WY Representation   is typically employed to pack a series rank-1 updates into a single compact representation [11]. We follow the formulation of $\mathbf{P}$ in Comba [39] to reduce the need for an additional matrix inversion in subsequent computations.

$$\mathbf{P}_{[t]}^r = \mathrm{Diag}(\boldsymbol{\gamma}_{[t]}^r) - \sum_{i=1}^{r} \mathrm{Diag}(\boldsymbol{\gamma}_{[t]}^{i\to r}) \boldsymbol{k}_{[t]}^i \boldsymbol{w}_{[t]}^{i\top} \qquad\qquad \mathbf{H}_{[t]}^r = \sum_{i=1}^{t} \mathrm{Diag}\left(\boldsymbol{\gamma}_{[t]}^{i\to r}\right) \boldsymbol{k}_{[t]}^i \boldsymbol{u}_{[t]}^{i\top} \tag{3}$$

where the auxiliary vector $\boldsymbol{w}_t \in \mathbb{R}^{d_k}$ and $\boldsymbol{u}_t \in \mathbb{R}^{d_v}$ are computed via the following recurrence relation:

$$\boldsymbol{w}_{[t]}^r = \beta_{[t]}^r \left(\mathrm{Diag}(\boldsymbol{\gamma}_{[t]}^r)\boldsymbol{k}_{[t]}^r - \sum_{i=1}^{r-1} \boldsymbol{w}_{[t]}^i \left(\boldsymbol{k}_{[t]}^{i\top} \mathrm{Diag}\left(\boldsymbol{\gamma}_{[t]}^{i\to r}\right) \boldsymbol{k}_{[t]}^r\right)\right) \tag{4}$$

$$\boldsymbol{u}_{[t]}^r = \beta_{[t]}^r \left(\boldsymbol{v}_{[t]}^r - \sum_{i=1}^{r-1} \boldsymbol{u}_{[t]}^i \left(\boldsymbol{k}_{[t]}^{i\top} \mathrm{Diag}\left(\boldsymbol{\gamma}_{[t]}^{i\to r}\right) \boldsymbol{k}_{[t]}^r\right)\right) \tag{5}$$

UT transform.   We apply the UT transform [45] to reduce non-matmul FLOPs, which is crucial to enable better hardware utilization during training.

$$\mathbf{M}_{[t]} = \left(\mathbf{I} + \mathrm{StrictTril}\left(\mathrm{Diag}\left(\beta_{[t]}\right)\left(\boldsymbol{\Gamma}_{[t]}^{1\to C} \odot \mathbf{K}_{[t]}\right)\left(\frac{\mathbf{K}_{[t]}}{\boldsymbol{\Gamma}_{[t]}^{1\to C}}\right)^\top\right)\right)^{-1} \mathrm{Diag}\left(\beta_{[t]}\right) \tag{6}$$

$$\mathbf{W}_{[t]} = \mathbf{M}_{[t]} \left(\boldsymbol{\Gamma}_{[t]}^{1\to C} \odot \mathbf{K}_{[t]}\right), \qquad \mathbf{U}_{[t]} = \mathbf{M}_{[t]} \mathbf{V}_{[t]} \tag{7}$$

The inverse of a lower triangular matrix can be efficiently computed through an iterative row-wise approach by forward substitution in Gaussian elimination [28].

Equivalently, in matrix form, we can update the state in chunk-wise:

$$\mathbf{S}_{[t+1]} = \mathrm{Diag}(\boldsymbol{\gamma}_{[t]}^C)\mathbf{S}_{[t]} + \left(\boldsymbol{\Gamma}_{[t]}^{i\to C} \odot \mathbf{K}_{[t]}\right)^\top \left(\mathbf{U}_{[t]} - \mathbf{W}_{[t]}\mathbf{S}_{[t]}\right) \in \mathbb{R}^{d_k \times d_v} \tag{8}$$

4

During the output stage, we adopt an inter-block recurrent and intra-block parallel strategy to maximize matrix multiplication throughput, thereby fully utilizing the computational potential of Tensor Cores.

$$\mathbf{O}_{[t]} = \underbrace{\left(\boldsymbol{\Gamma}_{[t]}^{1\rightarrow C} \odot \mathbf{Q}_{[t]}\right)\mathbf{S}_{[t]}}_{\text{inter chunk}} + \underbrace{\text{Tril}\left(\left(\boldsymbol{\Gamma}_{[t]}^{1\rightarrow C} \odot \mathbf{Q}_{[t]}\right)\left(\frac{\mathbf{K}_{[t]}}{\boldsymbol{\Gamma}_{[t]}^{1\rightarrow C}}\right)^{\top}\right)\underbrace{\left(\mathbf{U}_{[t]} - \mathbf{W}_{[t]}\mathbf{S}_{[t]}\right)}_{\text{"pseudo"-value term}}}_{\text{intra chunk}} \in \mathbb{R}^{C \times d_v} \tag{9}$$

3.2 Efficiency Analysis

In terms of representational capacity, KDA aligns with the generalized DPLR formulation, i.e., $\mathbf{S}_t = (\mathbf{D} - \boldsymbol{a}_t\boldsymbol{b}_t^{\top})\mathbf{S}_{t-1} + \boldsymbol{k}_t\boldsymbol{v}_t^{\top}$, both exhibiting fine-grained decay behavior. However, such fine-grained decay introduces numerical precision issues during division operations (e.g., the intra-chunk computation in Eq. 9). To address this, prior work such as GLA [113] performs computations in the logarithmic domain and introduces secondary chunking in FP32 precision. This approach, however, prevents full utilization of half-precision matrix multiplications and significantly reduces operator speed. by binding both variables $\boldsymbol{a}$ and $\boldsymbol{b}$ to $\boldsymbol{k}$, KDA effectively alleviates this bottleneck—reducing the number of second-level chunk matrix computations from four to two, and further eliminating three additional matrix multiplications. As a result, the operator efficiency of KDA improves by roughly 100% compared to the DPLR formulation. A detailed analysis is provided in §6.2.
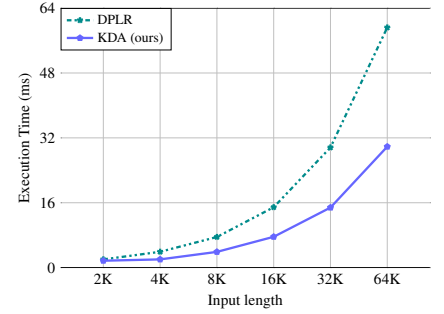

Figure 2: Execution time of kernels for varying input lengths, with a uniform batch size of 1 and 16 heads.

4 The Kimi Linear Model Architecture

The main backbone of our model architecture follows Moonlight [61]. In addition to fine-grained gating, we also leverage several components to further improve the expressiveness of Kimi Linear. The overall Kimi Linear architecture is shown in Figure 3.

Neural Parametarization Let $\boldsymbol{x}_t \in \mathbb{R}^d$ be the t-th token input representation, the input to KDA for each head h is obtained by

$$\boldsymbol{q}_t^h, \boldsymbol{k}_t^h = \text{L2Norm}(\text{Swish}(\text{ShortConv}(\mathbf{W}_{q/k}^h \boldsymbol{x}_t))) \in \mathbb{R}^{d_k}$$

$$\boldsymbol{v}_t^h = \text{Swish}(\text{ShortConv}(\mathbf{W}_v^h \boldsymbol{x}_t)) \in \mathbb{R}^{d_v}$$

$$\boldsymbol{\alpha}_t^h = f(\mathbf{W}_{\alpha}^{\uparrow}\mathbf{W}_{\alpha}^{\downarrow}\boldsymbol{x}_t) \in [0,1]^{d_k}$$

$$\beta_t^h = \text{Sigmoid}(\mathbf{W}_{\beta}^h \boldsymbol{x}_t) \in [0,1]$$

where $d_k, d)v$ represent the key and value head dimensions, which are set to 128 for all experiments. For $\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v}$, we apply a ShortConv followed by a Swish activation, following [110]. The $\boldsymbol{q}$ and $\boldsymbol{k}$ representations are further normalized using L2Norm to ensure eigenvalues stability, as suggested by [111]. The per-channel decay $\boldsymbol{\alpha}_t^h$ is parameterized via a low-rank projection and a decay function $f(\cdot)$ similar to those used in GDN and Mamba [110, 16]. Before the output projection through $\mathbf{W}_o \in \mathbb{R}^{d \times d}$, we use a head-wise RMSNorm [121] and a date-dependent gating mechanism [78] parameterized as:

$$\boldsymbol{o}_t = \mathbf{W}_o\left(\text{Sigmoid}\left(\mathbf{W}_g^{\uparrow}\mathbf{W}_g^{\downarrow}\boldsymbol{x}_t\right) \odot \text{RMSNorm}\left(\text{KDA}\left(\boldsymbol{q}_t, \boldsymbol{k}_t, \boldsymbol{v}_t, \boldsymbol{\alpha}_t, \beta_t\right)\right)\right) \tag{10}$$
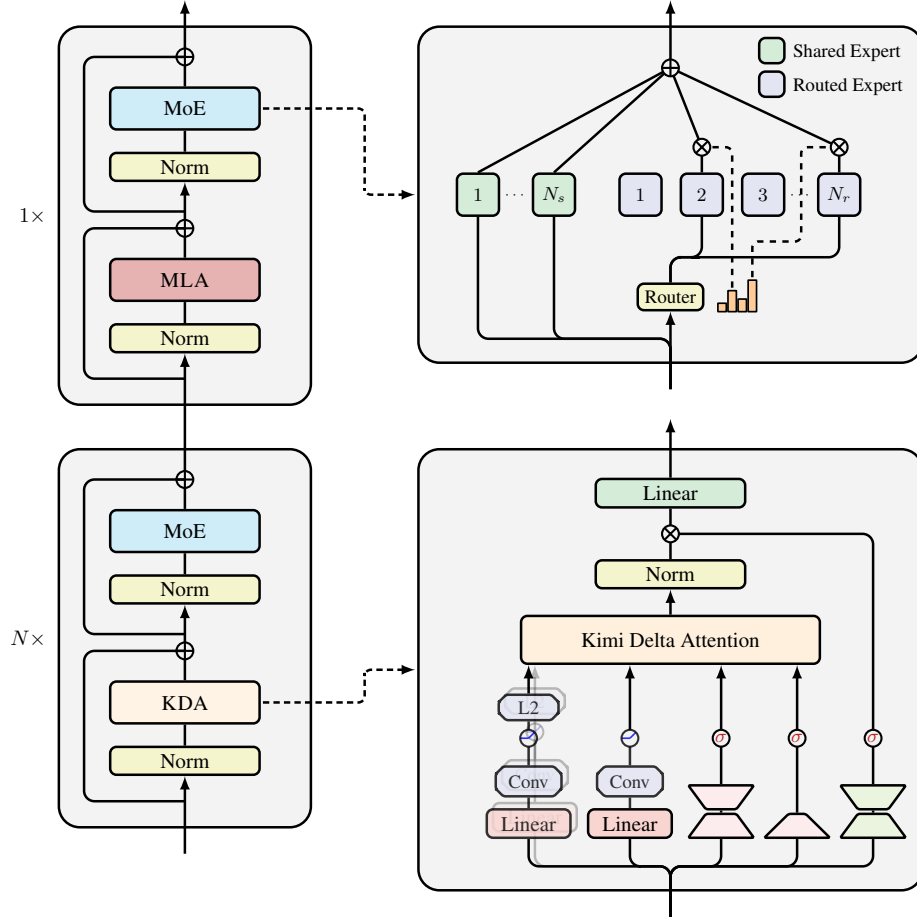
Figure 3: Illustration of our Kimi Linear model architecture, which consists of a stack of blocks containing a token mixing layer followed by a MoE channel-mixing layer. Specifically, we interleave N KDA layers with one MLA layer for token mixing, where N is set to 3 in our implementation.

where the output gate adopts a low-rank parameterization (with a rank equal to the head dimension) to ensure a fair parameter comparison, while maintaining performance comparable to full-rank gating and alleviating the Attention Sink [78]. The choice of nonlinear activation function is further discussed in §5.2.

Hybrid model architecture   Long-context retrieval remains the primary bottleneck for pure linear attention, we therefore hybridize KDA with a small number of full global-attention (Full MLA) layers [19]. For Kimi Linear, we chose a layerwise approach (alternating entire layers) over a headwise one (mixing heads within layers) for its superior infrastructure simplicity and training stability. Empirically, a uniform 3:1 ratio, i.e., repeating 3 KDA layers to 1 full MLA layer, provided the best quality–throughput trade-off. We discuss other hybridization strategies in § 7.2.

No Position Encoding (NoPE) for MLA Layers.   In Kimi Linear, we apply NoPE to all full attention (MLA) layers. This design delegates the entire responsibility for encoding positional information and recency bias (see § 6.1) to the KDA layers. KDA is thus established as the primary position-aware operator, fulfilling a role analogous to, or arguably stronger than, auxiliary components like short convolutions [3] or SWA [75]. Our findings align with prior results [109, 7, 19], who similarly demonstrated that complementing global NoPE attention with a dedicated position-aware mechanism yields competitive long-context performance.

We note that NoPE offers practical advantages, particularly for MLA. First, NoPE enables their conversion to the highly-efficient pure Multi-Query Attention (MQA) during inference. Second, it simplifies long-context training, as it obviates the need for RoPE parameter adjustments, such as frequency base tuning or methods like YaRN [71].
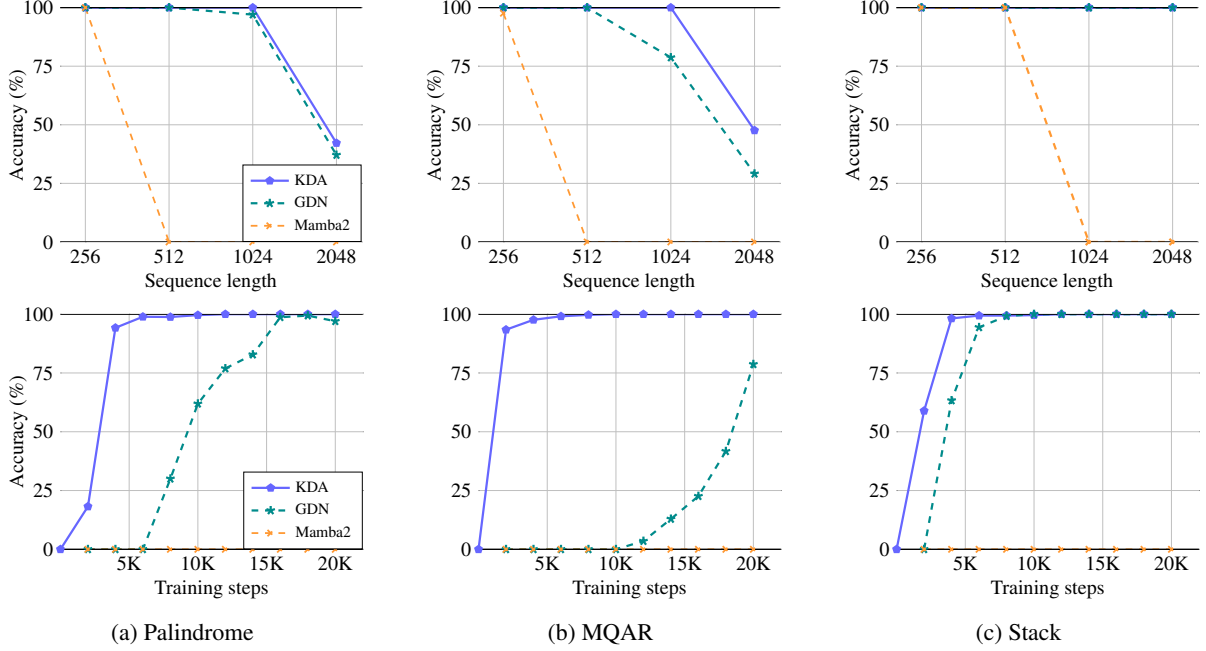
(a) Palindrome   (b) MQAR   (c) Stack

Figure 4: Results on synthetic tasks: palindrome, multi query associative recall, and the state tracking.

5 Experiments

5.1 Synthetic tests

We start by evaluating KDA against other competing linear attention methods on three synthetic tasks, serving as benchmark tests for long-context performance. Across all experiments, we adopt a consistent model configuration of 2 layers with 2 attention heads, each having a head dimension of 128. For each task, we train the model for at most 20,000 steps with a grid search over learning rates in $\{5 \times 10^{-5}, 1 \times 10^{-4}, 5 \times 10^{-4}, 1 \times 10^{-3}\}$. We then present the best-performing training accuracy curves. Specifically, we compare two scenarios: (1) the performance of different tasks as training length increases from 256 to 2,048 tokens, measuring the peak accuracy; and (2) the convergence speed of KDA, GDN, and Mamba2 with a fixed context length of 1,024 tokens.

Palindrome   Palindrome requires the model to reproduce a given sequence of random tokens in reverse order. As illustrated in Table 5.1, given an input like "O G R S U N E", the model must generate its exact reversal. Such copying tasks are known to be difficult for linear attention models [44], as they struggle to precisely retrieve the entire history from a compressed, fixed-size memory state.

Input	O	G	R	S	U	N	E	`<SEP>`	E	N	U	S	R	G	O
Output	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	N	U	S	R	G	O	ϕ

Multi Query Associative Recall (MQAR)   MQAR assesses the model's ability to retrieve values associated with multiple queries that appear at various positions within the context. For instance, as shown in Table 5.1, the model is asked to recall 0 for the query B and 5 for G. This task is known to be highly correlated with language modeling performance [5].

Input	A	1	C	3	B	0	M	8	G	5	E	4	`<SEP>`	B	G
Output	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	0	5

Stack   We assess the state tracking capabilities [27] of each candidate by simulating the standard LIFO (Last In First Out) stack operations. Our setup involves 64 independent stacks, each identified by a unique ID. The model processes a sequence of two operations: 1) PUSH: an action like "`<PUSH>` 1 G" adds the element G to stack 1; 2) POP: an action like "`<POP>` 0 E" requires the model to predict the element E most recently pushed onto stack 0. The objective is to accurately track the states of all stacks and predict the correct element upon each pop request.

Table 1: Ablation study on the hybrid ratio of KDA to MLA attention and other key components. We list the training and validation perplexities (lower is better) for comparison. The best-performing model, used in our final experiments, is highlighted in gray.

		Training PPL ($\downarrow$)	Validation PPL ($\downarrow$)
	3:1	**9.23**	**5.65**
	0:1	9.45	5.77
Hybrid ratio	1:1	9.29	5.66
	7:1	9.23	5.70
	15:1	9.34	5.82
w/o output gate		9.25	5.67
w/ swish output gate		9.43	5.81
w/o convolution layer		9.29	5.70

Figure 4 shows the final results. Across all tasks, KDA consistently achieves the highest accuracy as the sequence length increases from 256 to 2,048 tokens. In particular, on the Palindrome and recall-intensive MQAR tasks, KDA converges significantly faster than GDN. This confirms the benefits of our fine-grained decay, which enables the model to selectively forget irrelevant information while preserving crucial memories more precisely. We also observe that Mamba2 [16], a typical linear attention that uses only multiplicative decay and lacks a delta rule, fails on all tasks in our model settings.

5.2 Ablation on Key Components of Kimi Linear

We conducted a series of ablation studies by directly comparing different models to the first-scale scaling law model, i.e., 16 heads, 16 layers. All models were trained with the same FLOPs budget and hyperparameters for a fair comparison. We report the training and validation perplexities (PPLs) in Table 1. The validation PPL is calculated on a high-quality dataset whose distribution differs significantly from the pre-training corpus, emphasizing generalization under distribution shift, and thus the differences in training and validation perplexities.

Output gate   We compare our default Sigmoid output gate against two variants: one with no gating and another with swish gating. The results show that removing the gate degrades performance. Moreover, the swish gate adopted by [110] performs substantially worse than Sigmoid. Our observation is consistent with [78], who also conclude that Sigmoid gating offers superior performance. So we adopt Sigmoid across all of our experiments, including GDN-H.

Convolution Layer   Lightweight depthwise convolutions with a small kernel size (e.g., 4) can be effective at capturing local token dependencies [3] and are widely adopted by many recent architectures [16, 5, 111]. We validate its efficacy in Table 1, demonstrating that convolutional layers continue to play a non-negligible role in hybrid models.

Hybrid ratio   We performed an ablation study to determine the optimal hybrid ratio of KDA linear attention layers to MLA full attention layers. Among the configurations tested, the 3:1 ratio (3 KDA layers for every 1 MLA layer) yielded the best results, achieving the lowest training and validation losses. We observed clear trade-offs with other ratios: a higher ratio (e.g., 7:1) produced a comparable training loss but led to significantly worse validation performance, while a lower ratio (e.g., 1:1) maintained a similar validation loss but at the cost of increased inference overhead. Furthermore, the pure full-attention baseline (0:1) performed poorly. Thus, the 3:1 configuration offers the most effective balance between model performance and computational efficiency.

NoPE v.s. RoPE   As shown in Table 5, the Kimi Linear consistently excels on long-context evaluations, whereas Kimi Linear (RoPE) attains similar scores on short-context tasks. We posit that this divergence arises from how positional bias is distributed across depth. In Kimi Linear (RoPE), the global attention layer carries a strong, explicit relative positional signal, while the linear attention (e.g., GDN) contributes a weaker, implicit positional inductive bias. This mismatch yields an overemphasis on short-range order in the global layer, which benefits short contexts but makes the model less flexible when adapting mid-training to extended contexts. By contrast, Kimi Linear induces a more balanced positional bias across layers, which improves robustness and extrapolation at long ranges, leading to stronger long-context performance. Regarding long context performance, as shown in Table 5, Kimi Linear achieves the best average score across different long context benchmarks, which verifies the benefits we claim in the last section.

Table 2: Model configurations and hyperparameters for scaling law experiments.

# Act. Params.[†]	Head	Layer	Hidden	Tokens	lr	batch size[‡]
653M	16	16	1216	38.8B	2.006×10^{-3}	336
878M	18	18	1376	59.8B	1.790×10^{-3}	432
1.1B	20	20	1536	85.2B	1.617×10^{-3}	512
1.4B	22	22	1632	102.5B	1.486×10^{-3}	576
1.7B	24	24	1776	128.0B	1.371×10^{-3}	640

[†] Denotes the number of activated parameters in our MoE models, excluding embeddings.
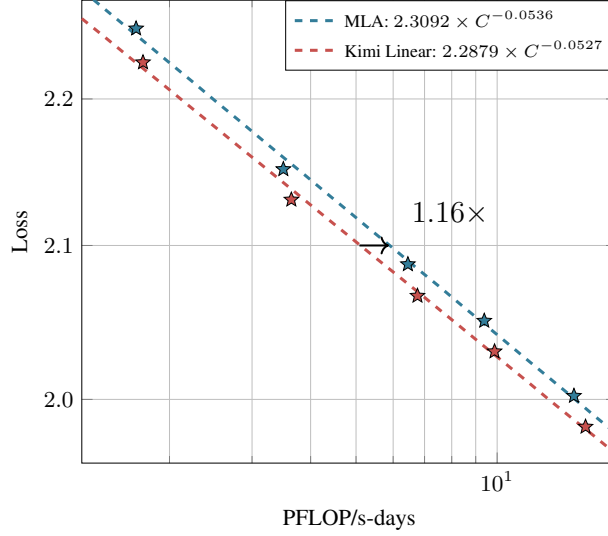[‡] All models were trained with a context length of 4,096.

Figure 5: The fitted scaling law curves for MLA and Kimi Linear.

5.3 Scaling Law of Kimi Linear

We conducted scaling law experiments on a series of MoE models following the Moonlight [61] architecture. In all experiments, we activated 8 out of 64 experts and utilized the Muon optimizer [61]. Details and hyperparameters are listed in Table 2.

For MLA, following the Chinchilla scaling law methodology [36], we trained five language models of different sizes, carefully tuning their hyperparameters through grid search to ensure optimal performance for each model. For KDA, we maintained the best hybrid ratio of 3:1 as ablated in Table 1. Except for this, we adhered strictly to the MLA training configuration without any modifications. As shown in Figure 5, Kimi Linear achieves $\sim 1.16\times$ computational efficiency compared to the MLA baselines with compute optimal training. We expect that careful hyperparameter tuning will yield superior scaling curves for KDA.

5.4 Experimental Setup

Kimi Linear and baselines settings   We evaluate our Kimi Linear model against a full-attention MLA baseline and a hybrid Gated DeltaNet (GDN-H) baseline, all of which share the same architecture, parameter count, and training setup for fair comparisons. The model configuration is largely aligned with Moonlight [61], with the key distinction that MoE sparsity is increased to 32. Each model activates 8 out of 256 experts, including one shared expert, resulting in 48 billion total parameters and 3 billion active parameters per forward pass. The first layer is implemented as a dense layer without MoE, ensuring stable training. To evaluate the effectiveness of NoPE in Kimi Linear, we also introduce a hybrid KDA baseline using RoPE with the same model configuration, referred to as Kimi Linear (RoPE).

Evaluation Benchmarks   Our evaluation encompasses three primary categories of benchmarks, each designed to assess distinct capabilities of the model:

- **Language Understanding and Reasoning**: Hellaswag [120], ARC-Challenge [14], Winogrande [82], MMLU [35], TriviaQA [46], MMLU-Redux [26], MMLU-Pro [102], GPQA-Diamond [81], BBH [93], and LiveBench [104].
- **Code Generation**: LiveCodeBench v6 [4][43], EvalPlus [59].
- **Math & Reasoning**: AIME 2025, MATH 500, HMMT 2025, PolyMath-en.
- **Long-context**: MRCR [5] , RULER [37], Frames [51], HELMET-ICL [117], RepoQA [60], Long Code Arena [13] and LongBench v2 [6].
- **Chinese Language Understanding and Reasoning**: C-Eval [42], and CMMLU [54].

Evaluation Configurations   All models are evaluated using temperature 1.0. For benchmarks with high variance, we report the score of Avg@k. For base model, We employ perplexity-based evaluation for MMLU, MMLU-Redux, GPQA-Diamond, and C-Eval. Otherwise, generation-based evaluation is adopted. To mitigate the high variance inherent to GPQA-Diamond, we report the mean score across eight independent runs. All evaluations are conducted using our internal framework derived from LM-Harness-Evaluation [10], ensuring consistent settings across all models.

5.4.1 Pretraining recipe

Pretraining recipe   All models are pretrained using a 4,096-token context window, the MuonClip optimizer, and the WSD learning rate schedule, processing a shared total of 1.4 trillion tokens sampled from the K2 pretraining corpus [49]. The learning rate is set to 1.1×10^{-3}, and the global batch size is fixed at 32 million tokens. They also adopt the same annealing schedule and long-context activation phase established in Kimi K2 [49].

Our final released Kimi Linear checkpoint is pretrained using the same procedure, but with an expanded total of 5.7 trillion tokens to match the pretraining tokens of Moonlight. In addition, the final checkpoint supports a context length of up to 1 million tokens. We compare the performance of Kimi Linear@5.7T and Moonlight in Appendix D

5.4.2 Posttraining recipe

SFT recipe   The SFT dataset extends the Kimi K2 [49] SFT data by incorporating additional reasoning tasks, creating a large-scale instruction-tuning dataset that spans diverse domains with a heavy emphasis on math and coding. We employ a multi-stage SFT approach, initially training the model on a broad range of diverse SFT data for general instruction-following, followed by scheduled targeted training on reasoning-intensive data to enhance the model's reasoning capabilities.

RL recipe   For the RL training prompt set, we primarily integrate three data sources: mathematics, code, and STEM. The main purpose of this enhancement is to boost the model's reasoning ability. Before conducting RL, we pre-selected data that matches a moderate difficulty level for the starting checkpoint.

A known risk of RL training is the potential degeneration of general capabilities. To mitigate this, we incorporate the PTX loss [69] during RL, following the practice of K2 [49]. This involves concurrent SFT on a high-quality, distributionally diverse dataset in the RL progress. Our PTX dataset spans both reasoning and general-purpose tasks. All data mentioned above are subsets derived from the training recipe of the K2 model [49].

For the RL algorithm, we use the same algorithm as in K1.5 [94], while introducing serval advanced tricks. We noticed that the precision mismatch between training and inference engines may lead to unstable RL learning. Therefore we introduce truncated importance sampling, a method that effectively mitigates the policy mismatch between rollout and training [115]. We also dynamically adjust the KL penalty and the mini batch size (*i.e.*, the number of updates per iteration) to make the RL training stable and avoid collapse of entropy [15].

5.5 Main results

5.5.1 Kimi Linear@1.4T results

Pretrain results   We compared our Kimi Linear model against two baselines (MLA and hybrid GDN-H) using a 1.4T pretraining corpus in Table 3. The evaluation focused on three areas: general knowledge, reasoning (math and code), and Chinese tasks. Kimi Linear consistently outperformed both baselines across almost all categories.

- General Knowledge: Kimi Linear scores highest on all of the key benchmarks like BBH, MMLU and HellaSwag.

---

[4]Questions from 2024.8 to 2025.5
[5]https://huggingface.co/datasets/openai/mrcr

- Reasoning: It leads in math (GSM8K) and most code tasks (CRUXEval). However, it scores slightly lower on EvalPlus compared to GDN-H.
- Chinese Tasks: Kimi Linear achieves the top scores on CEval and CMMLU.

In summary, Kimi Linear demonstrated the strongest performance, positioning it as a strong alternative to full-attention architectures at short context pretraining.

Table 3: Performance comparison of Kimi Linear with the full-attention MLA baseline and the hybrid GDN baseline, all using the same SFT recipe after pretraining. Kimi Linear consistently outperforms both MLA and GDN-H on short-context instruction-tuned benchmarks. Best per-column results are **bolded**.

		MLA	GDN-H	Kimi Linear
	Trained Tokens	1.4T	1.4T	1.4T
General	HellaSwag	81.7	82.2	**82.9**
	ARC-challange	64.6	66.5	**67.3**
	Winogrande	78.1	77.9	**78.6**
	BBH	71.6	70.6	**72.9**
	MMLU	71.6	72.2	**73.8**
	MMLU-Pro	47.2	47.9	**51.0**
	TriviaQA	68.9	70.1	**71.7**
Math & Code	GSM8K	83.7	81.7	**83.9**
	MATH	**54.7**	54.1	**54.7**
	EvalPlus	59.5	**63.1**	60.2
	CRUXEval-I-cot	51.6	56.0	**56.6**
	CRUXEval-O-cot	61.5	58.1	**62.0**
Chinese	CEval	79.3	79.1	**79.5**
	CMMLU	79.5	80.7	**80.8**

Table 4: Performance comparison of Kimi Linear with the full-attention MLA baseline and the hybrid GDN baseline, all using the same SFT recipe after pretraining. Kimi Linear consistently outperforms both MLA and GDN-H on short-context instruction-tuned benchmarks. Best per-column results are **bolded**.

		MLA	GDN-H	Kimi Linear
	Trained Tokens	1.4T	1.4T	1.4T
General	BBH	68.2	68.5	**69.4**
	MMLU	75.7	75.6	**77.0**
	MMLU-Pro	65.7	64.8	**67.4**
	MMLU-Redux	79.2	78.7	**80.3**
	GPQA-Diamond (Avg@8)	57.1	58.6	**62.1**
	Livebench (Pass@1)	45.7	**46.4**	45.2
Math & Code	AIME 2025 (Avg@64)	20.6	21.1	**21.3**
	MATH500 (Acc.)	80.8	**83.0**	81.2
	HMMT 2025 (Avg@32)	11.3	11.3	**12.5**
	PolyMath-en (Avg@4)	41.3	41.5	**43.6**
	LiveCodeBench v6 (Pass@1)	25.1	25.4	**26.0**
	EvalPlus	**62.6**	62.5	61.0

SFT results   Kimi Linear demonstrates strong performance across both general and math & code tasks after undergoing the same supervised fine-tuning (SFT) recipe, consistently outperforming MLA and GDN-H. In general tasks, Kimi Linear leads across the board, achieving the top scores on various MMLU benchmarks, BBH, and GPQA-Diamond. In math & code tasks, it surpasses both baselines on difficult benchmarks like AIME 2025, HMMT 2025, PolyMath-en, and LiveCodeBench. Despite some minor exceptions like MATH500 and EvalPlus, Kimi Linear shows robust superiority across the tasks, confirming its clear superiority to the other models tested (GDN-H and MLA).

Long Context Performance Evaluation   We evaluate the long-context performance of Kimi Linear against three baseline models—MLA, GDN-H, and Kimi Linear (RoPE)—across several benchmarks at 128k context length (see

Table 5: Comparisons of Kimi Linear with MLA, GDN-H, and Kimi Linear (RoPE) across long-context benchmarks. The last column reports the overall average (↑). All models is trained on 1T tokens. Best per-column results are **bolded**.

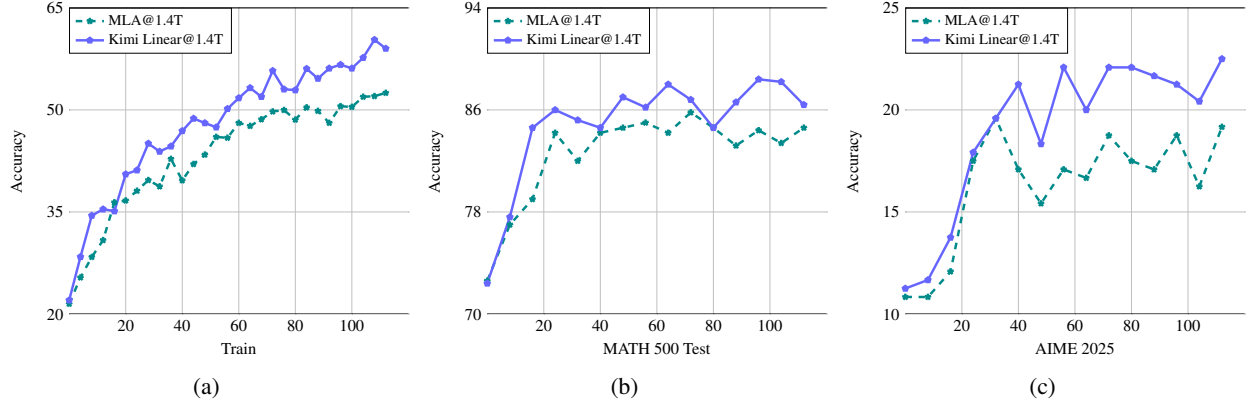|	RULER	MRCR	HELMET-ICL	LongBench V2	Frames	RepoQA	Long Code Arena		Avg.
							Lib	Commit	
MLA	81.3	22.6	88.0	**36.1**	**60.5**	63.0	32.8	**33.2**	52.2
GDN-H	80.5	23.9	85.5	32.6	58.7	63.0	34.7	30.5	51.2
Kimi Linear (RoPE)	78.8	22.0	88.0	35.4	59.9	66.5	31.3	32.5	51.8
Kimi Linear	**84.3**	**29.6**	**90.0**	35.0	58.8	**68.5**	**37.1**	32.7	**54.5**

Figure 6: The training and test accuracy curves for Kimi Linear@1.4T and MLA@1.4T during Math RL training. Kimi Linear consistently outperforms the full attention baseline by a sizable margin during the whole RL process.

Table 5). The results highlight Kimi Linear's clear superiority in these long-context tasks. It consistently outperformed MLA and GDN-H, achieving the highest scores on RULER (84.3) and RepoQA (68.5) by a significant margin. This pattern of outperformance held across most other tasks, except for LongBench V2 and Frames. Overall, Kimi Linear achieved the highest average score (54.5), further reinforcing its effectiveness as a leading attention architecture in long-context scenarios.

RL results   To compare the RL convergence properties of Kimi Linear and MLA, we conduct RLVR using the in-house mathematics training set from [49], and evaluate on mathematics test sets (e.g., AIME 2025, MATH500), while keeping the algorithm and all hyperparameters identical to ensure a fair comparison of performance.

As shown in Figure 6, Kimi Linear demonstrates better efficiency compared to MLA. On the training set, even though both models start at similar points, the growth rate of training accuracy for Kimi Linear is significantly higher than that of MLA, and the gap gradually widens. On the test set, similar phenomena are observed. For example, on MATH500 and AIME2025, Kimi Linear achieves faster and better improvement compared to MLA. Overall, in reasoning-intensive long-form generation under RL, we empirically observe that Kimi Linear performs significantly better than MLA.

Summary of overall findings   During the pretraining and SFT stages, a clear performance hierarchy was established: Kimi Linear outperformed GDN-H, which in turn outperformed MLA. However, this hierarchy shifted in long-context evaluations. While Kimi Linear maintained its top position, GDN-H's performance declined, placing it behind MLA. Furthermore, in the RL stage, Kimi Linear also demonstrated superior performance over MLA. Overall, Kimi Linear consistently ranked as the top performer across all stages, establishing itself as a superior alternative to full attention architectures.

5.6   Efficiency Comparison

Prefilling & Decoding speed   We compare the training and decoding times for full attention MLA [19], GDN-H, and Kimi Linear in Figure 7a and Figure 7b. Note that all models are based on the Kimi Linear 48B setting, with the same number of layers and attention heads. We observe that: 1) Despite incorporating a more fine-grained decay mechanism, Kimi Linear introduces negligible latency overhead compared to GDN-H during prefilling. As shown in Figure 7a, their performance curves are virtually indistinguishable, confirming that our method maintains high efficiency. The hybrid Kimi Linear model demonstrates a clear efficiency advantage over the MLA baseline as sequence length increases.
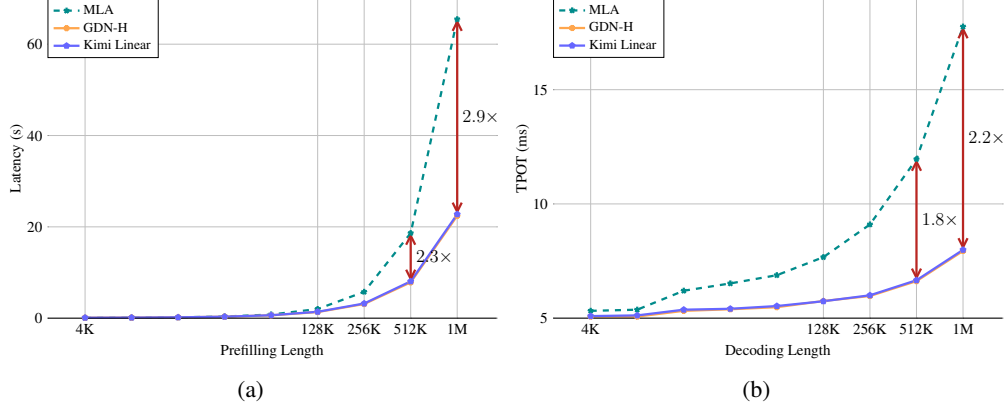
Figure 7: (a) The prefilling time of MLA (full attention), hybrid GDN-H and our KDA-H. (b) The time per output token (TPOT) for MLA, GDN-H and KDA-H during decoding. (We use batch size = 1 here for tests.)

While its performance is comparable to MLA at shorter lengths (4k–16k), it becomes significantly faster from 128k onwards. This efficiency gap widens dramatically at scale, with Kimi Linear outperforming MLA by a factor of 2.3 for 512k sequences and 2.9 for 1M sequences. 3) Kimi Linear fully demonstrates its advantages during the decoding phase. For decoding at 1M context length, Kimi Linear is $6\times$ faster than full attention.

6   Discussions

6.1   Kimi Delta Attention as learnable position embeddings

The standard attention in transformers is by design agnostic to the sequence order of its inputs [98], thus necessitating explicit positional encodings [74, 85]. Among various methods, RoPE [87] has emerged as the *de facto* standard in modern LLMs due to its effectiveness [97, 1, 19]. The mechanism of multiplicative positional encodings like RoPE can be analyzed through a generalized attention formulation:

$$s_{t,i} = \boldsymbol{q}_t^\top \left(\prod_{j=i+1}^{t} \mathbf{R}_j \right) \boldsymbol{k}_i \tag{11}$$

where the position relationship between the t-th query $\boldsymbol{q}_t$ and the i-th key $\boldsymbol{k}_i$ is reflected by the cumulative matrix products. RoPE defines the transformation matrix $\mathbf{R}_j$ as a block diagonal matrix composed of $d_k/2$ 2D rotation matrices $\mathbf{R}_j^k = \begin{pmatrix} \cos(j\theta_k) & -\sin(j\theta_k) \\ \sin(j\theta_k) & \cos(j\theta_k) \end{pmatrix}$ with **per-2-dimensional** angular frequency θ_k. Due to the properties of rotation matrices, i.e., $\mathbf{R}_{t-i} = \mathbf{R}_t^\top \mathbf{R}_i$, absolute positional information $\mathbf{R}_t$ and $\mathbf{R}_i$ can be applied separately to $\boldsymbol{q}_t$ and $\boldsymbol{k}_i$, which are then transformed into relative positional information $t-i$ encoded as $\prod_{j=i+1}^{t} \mathbf{R}_j = \begin{pmatrix} \cos((t-i)\theta_k) & -\sin((t-i)\theta_k) \\ \sin((t-i)\theta_k) & \cos((t-i)\theta_k) \end{pmatrix}$.

Consequently, we show that linear attentions with the gated delta rule can be expressed in a comparable formulation in Eq. 12. Similar forms for other attention variants are summarized in Table 6.

$$\boldsymbol{o}_t = \sum_{i=1}^{t} \left(\boldsymbol{q}_t^\top \left(\prod_{j=i+1}^{t} \mathbf{A}_j \left(\mathbf{I} - \beta_j \boldsymbol{k}_j \boldsymbol{k}_j^\top \right) \right) \boldsymbol{k}_j \right) \boldsymbol{v}_j \tag{12}$$

From this perspective, GDN can be interpreted as a form of multiplicative positional encoding whose transition matrix is data-dependent, thereby relaxing the orthogonality constraint imposed by RoPE and can be potentially more powerful [114]. [6] This provides a potential solution to the known extrapolation issues of RoPE, whose fixed frequencies can cause overfitting to context lengths seen during training [107, 71]. Some recent works adopt workarounds like partial RoPE [7] or even forgo explicit positional encodings entirely (NoPE) [48, 75, 19]. Given that GDN serves as an analogue role

---

[6]When preserving orthogonality, absolute positional encodings can be applied independently to $\boldsymbol{q}$ and $\boldsymbol{k}$, which are then automatically transformed into relative positional encodings during the attention computation [86].

Table 6: An overview of attention mechanisms in their mathematically equivalent recurrent (o_t) and parallel (**O**) forms. We omitted the normalization term and β_t to achieve a more concise representation.

	Recurrent form	Parallel form
SA [98]	$\sum_{j=1}^{t} \exp\left(q_t^\top k_j\right) v_j$	$\left(\exp\left(\mathbf{QK}^\top\right) \odot \mathbf{M}\right) \mathbf{V}$
SA + RoPE [87]	$\sum_{j=1}^{t} \exp\left(q_t^\top \left(\prod_{s=j+1}^{t} R_s\right) k_j\right) v_j$	$\left(\exp\left(\mathbf{R}\left(\mathbf{Q}\right) \mathbf{R}\left(\mathbf{K}\right)^\top\right) \odot \mathbf{M}\right) \mathbf{V}$
LA [100]	$\sum_{j=1}^{t} \left(q_t^\top k_j\right) v_j$	$\left(\mathbf{QK}^\top \odot \mathbf{M}\right) \mathbf{V}$
Mamba2 [16]	$\sum_{j=1}^{t} \left(q_t^\top \left(\prod_{s=j+1}^{t} \alpha_s\right) k_j\right) v_j$	$\left(\mathbf{QK}^\top \odot \mathcal{A} \odot \mathbf{M}\right) \mathbf{V}$
GLA [113]	$\sum_{j=1}^{t} \left(q_t^\top \left(\prod_{s=j+1}^{t} \mathrm{Diag}\left(\alpha_s\right)\right) k_j\right) v_j$	$\left(\left(\mathbf{Q} \odot \mathbf{\Gamma}\right) \left(\frac{\mathbf{K}}{\mathbf{\Gamma}}\right) \odot \mathbf{M}\right) \mathbf{V}$
DeltaNet [111]	$\sum_{j=1}^{t} \left(q_t^\top \left(\prod_{s=j+1}^{t} \left(\mathbf{I} - k_s k_s^\top\right)\right) k_j\right) v_j$	$\left(\mathbf{QK}^\top \odot \mathbf{M}\right) \left(\mathbf{I} + \mathbf{KK}^\top \odot \mathbf{M}^-\right)^{-1} \mathbf{V}$
FoX [57]	$\sum_{j=1}^{t} \exp\left(q_t^\top \left(\prod_{s=j+1}^{t} \alpha_s\right) k_j\right) v_j$	$\left(\exp\left(\mathbf{QK}^\top\right) \odot \mathcal{A} \odot \mathbf{M}\right) \mathbf{V}$
DeltaFormer [124]	$\sum_{j=1}^{t} \exp\left(q_t^\top \left(\prod_{s=j+1}^{t} \left(\mathbf{I} - \exp\left(k_s k_s^\top\right)\right)\right) k_j\right) v_j$	$\left(\exp\left(\mathbf{QK}^\top\right) \odot \mathbf{M}\right) \left(\mathbf{I} + \exp\left(\mathbf{KK}^\top\right) \odot \mathbf{M}^-\right)^{-1} \mathbf{V}$
PaTH [114]	$\sum_{j=1}^{t} \exp\left(q_t^\top \left(\prod_{s=j+1}^{t} \alpha_s \left(\mathbf{I} - k_s k_s^\top\right)\right) k_j\right) v_j$	$\left(\exp\left(\left(\mathbf{QK}^\top \odot \mathbf{M}\right) \left(\mathbf{I} + \mathbf{KK}^\top \odot \mathbf{M}^-\right)^{-1} \odot \mathcal{A} \odot \mathbf{M}\right)\right) \mathbf{V}$
GDN [110]	$\sum_{j=1}^{t} \left(q_t^\top \left(\prod_{s=j+1}^{t} \alpha_s \left(\mathbf{I} - k_s k_s^\top\right)\right) k_j\right) v_j$	$\left(\mathbf{QK}^\top \odot \mathcal{A} \odot \mathbf{M}\right) \left(\mathbf{I} + \mathbf{KK}^\top \odot \mathcal{A} \odot \mathbf{M}^-\right)^{-1} \mathbf{V}$
Comba [39]	$\sum_{j=1}^{t} \left(q_t^\top \left(\prod_{s=j+1}^{t} \left(\alpha_s - k_s k_s^\top\right)\right) k_j\right) v_j$	$\left(\mathbf{QK}^\top \odot \mathcal{A} \odot \mathbf{M}\right) \left(\mathbf{I} + \mathbf{KK}^\top \odot \mathcal{A}^{i-1/j} \odot \mathbf{M}^-\right)^{-1} \mathbf{V}$
RWKV7 [70]	$\sum_{j=1}^{t} \left(q_t^\top \left(\prod_{s=j+1}^{t} \left(\mathrm{Diag}\left(\alpha_s\right) - k_s k_s^\top\right)\right) k_j\right) v_j$	$\left(\left(\mathbf{Q} \odot \mathbf{\Gamma}\right) \left(\frac{\mathbf{K}}{\mathbf{\Gamma}}\right)^\top \odot \mathbf{M}\right) \left(\mathbf{I} + \left(\mathbf{K} \odot \overset{0 \to t-1}{\mathbf{\Gamma}}\right) \left(\frac{\mathbf{K}}{\mathbf{\Gamma}}\right)^\top \odot \mathbf{M}^{-1}\right)^{-1} \mathbf{V}$
KDA (ours)	$\sum_{j=1}^{t} \left(q_t^\top \left(\prod_{s=j+1}^{t} \mathrm{Diag}\left(\alpha_s\right) \left(\mathbf{I} - k_s k_s^\top\right)\right) k_j\right) v_j$	$\left(\left(\mathbf{Q} \odot \mathbf{\Gamma}\right) \left(\frac{\mathbf{K}}{\mathbf{\Gamma}}\right)^\top \odot \mathbf{M}\right) \left(\mathbf{I} + \left(\mathbf{K} \odot \mathbf{\Gamma}\right) \left(\frac{\mathbf{K}}{\mathbf{\Gamma}}\right)^\top \odot \mathbf{M}^{-1}\right)^{-1} \mathbf{V}$

to RoPE, we choose NoPE for global full attention layers (MLA) in our model, allowing positional information to be captured dynamically by our proposed KDA model.

Moreover, a key strength of RoPE is its fine-grained positional encoding, achieved by assigning different rotation frequencies to each pair of dimensions, which functions analogously to a Nonuniform Fourier Transform [7, 40] along the feature dimension. Standard GDN, however, employs a per-head scalar decay and lacks this per-dimensional diversity, which motivates us to propose KDA with a learnable channel-wise gate.

6.2   Relation to DPLR

We can rewrite Eq. 1 as $\mathbf{S}_t = \left(\mathrm{Diag}(\alpha_t) - \beta_t k_t k_t^\top \mathrm{Diag}(\alpha_t)\right) \mathbf{S}_{t-1} + \beta_t k_t v_t^\top$. This formulation is equivalent to a constrained variant of the DPLR update rule, with the correspondence between the two given by:

$$\mathbf{S}_t = (\mathbf{D} - a_t b_t^\top)\mathbf{S}_{t-1} + k_t v_t^\top, \text{s.t.,} \quad \mathbf{D} = \mathrm{Diag}(\alpha_t), a_t = \beta_t k_t, b_t = k_t \odot \alpha_t.$$

Hence, KDA preserves the *Diagonal-Plus-Low-Rank* (DPLR) state transition structure inherent in the original formulation. Furthermore, by sharing α_t, we can factor it out as in Eq. 1, enabling a fine-grained multiplicative decay over $\mathbf{S}_t$ in a manner similar to GLA [113], followed by a Householder-style transformation like DeltaNet [83, 111] for efficient state updating.

We provide a side-by-side comparison of the chunkwise PyTorch-style pseudocode implementations for DPLR and KDA in Listing 8a and Listing 8b. The key improvements are highlighted below:

- Listing 8a Line 13-16 v.s., Listing 8b Line 14-15 : the reciprocal of the cumulative decay term $1/\Gamma$ in chunkwise form (Eq. 9) can introduce numerical instability. While we can resolve this issue by secondary chunking [112], it incurs additional computation and I/O overhead. By fixing $a = b = k$ in the DPLR formulation, KDA removes the need for two secondary chunking steps, substantially reducing redundant operations and improving overall efficiency.

```
1 def chunk_dplr(q, k, v, a, b, g, chunk_size):
2     B, H, T, K, V, BT = *q.shape, v.shape[-1], chunk_size
3     NT, S = T // BT, k.new_zeros(B, H, K, V)
4     q, k, v, a, b, g = map(lambda x: rearrange(x, 'b h (n c)
  ↪     d -> b h n c d', c=BT), [q, k, v, a, b, g])
5     gc = g.cumsum(-2)
6 -   Aab, Aak, Aqb, Aqk = (torch.zeros(B, H, NT, BT, BT) for
  ↪     _ in range(4))
7
8     for i in range(BT):
9         a_i, q_i, g_i = (x[:,:,:,i,None] for x in (a, q,
  ↪         gc))
10        mask = (torch.arange(BT) <= i)[..., None]
11        s1_i = (g_i - gc).exp().where(mask, 0)
12        s2_i= (g_i - g[:,:,:,i,None] - gc).where(mask, 0)
13 -      Aqk[..., i, :] = (q_i * k * s1_i).sum(-1)
14 -      Aqb[..., i, :] = (q_i * b * s1_i).sum(-1)
15 -      Aab[..., i, :] = (a_i * b * s2_i).sum(-1)
16 -      Aak[..., i, :] = (a_i * k * s2_i).sum(-1)
17    for i in range(1, BT):
18        Aab[..., i, :i] = Aab[..., i, :i] + (Aab[..., i, :,
  ↪         None] * Aab[..., :, :i]).sum(-2)
19    Aab = Aab + torch.eye(BT)
20    u, w = Aab @ (Aak @ v), Aab @ ((gc-g).exp() * a)
21    o = torch.zeros_like(v)
22    mask = torch.triu(torch.ones(BT, BT), diagonal=1)
23    for i in range(0, NT):
24        q_i, k_i, v_i, u_i, w_i, b_i = (x[:, :, i] for x in
  ↪         (q, k, v, u, w, b))
25 -      o1 = Aqk[:, :, i] @ v_i
26 -      o2 = Aqb[:, :, i] @ (u_i + w_i @ S)
27 -      o3 = (q_i * gkc[:, :, i].exp()) @ S
28        o[:, :, i] = o1 + o2 + o3
29        decay = (gc[:, :, i, -1, None] - gc[:, :, i]).exp()
30        S = S * gc[:,:,i,-1,:,None].exp()
31 -      S +=(k_i * decay).transpose(-1,-2) @ v_i
32 -      S +=(b_i * decay).transpose(-1,-2) @ (u_i + w_i @ S)
33    return o, S
```

(a) PyTorch-style pseudo code for chunkwise DPLR.

```
1 def chunk_KDA(q, k, v, a, b, g, chunk_size):
2     B, H, T, K, V, BT = *q.shape, v.shape[-1], chunk_size
3     NT, S = T // BT, k.new_zeros(B, H, K, V)
4     q, k, v, g = map(lambda x: rearrange(x, 'b h (n c) ...
  ↪     -> b h n c ...', c=BT), [q, k, v, g])
5     gc = g.cumsum(-2)
6 +   Aqk, Akk = (torch.zeros(B, H, NT, BT, BT) for _ in
  ↪     range(2))
7
8     for i in range(BT):
9         k_i, q_i = k[:, :, :, i, None], q[:, :, :, i, None]
10        g_i = gc[...,i:i+1,:]
11        mask = (torch.arange(BT) <= i)[..., None]
12        s1_i = (g_i - gc).exp().where(mask, 0)
13        s2_i = (gc - g_i).exp()
14 +      Aqk[:, :, :, i, :] = (q_i * k * s1_i).sum(-1)
15 +      Akk[..., i] = (k_i * k * s2_i).sum(-1)
16    mask = torch.triu(torch.ones(BT, BT), diagonal=0)
17    A = -Akk.masked_fill(mask, 0)
18    for i in range(1, BT):
19        A[..., i, :i] = A[..., i, :i] + (A[..., i, :, None]
  ↪         * A[..., :, :i].clone()).sum(-2)
20    A = (A + torch.eye(BT))
21    w, u = A @ (gc.exp() * k), A @ v
22    o = torch.zeros_like(v)
23    mask = torch.triu(torch.ones(BT, BT), diagonal=1)
24    for i in range(0, NT):
25        q_i, k_i, u_i, g_i, w_i = (x[:, :, i] for x in (q, k,
  ↪         u, gc, w))
26 +      o[:,:,i]=(q_i *g_i.exp()) @ S + Aqk @(u_i-w_i @ S)
27        decay = (g_i[:,:,-1:] - g_i).exp()
28        S = S * g_i[:, :, -1, :, None].exp()
29 +      S += (k_i * decay).transpose(-1,-2) @ v_i
30    return o, S
```

(b) PyTorch-style pseudo code for chunkwise KDA.

- Listing 8a Line 25-27,31-32 v.s., Listing 8b Line 26,29 : KDA further eliminates roughly three matrix multiplications during inter-chunk and output computation, leading to significant kernel-level acceleration.

We further benchmark the kernel speed in Fig. 2, showing that KDA achieves nearly $2\times$ speed than DPLR for sequence lengths up to $64k$.

6.3 Complexity Analysis

Training flops   We maintain a similar number of parameters in Kimi Linear as in the full attention MLA. The linear projection calculation remains identical to that of the global attention layer. The key distinction lies in the FLOPs associated with attention computation. For simplicity, we focus on non-variable length scenarios. Based on the implementation of the gated rule kernel, the theoretical FLOPs for a single attention head with headdim d_h and a fixed chunk size $C = 64$ in the gated delta rule [101] (per sequence of length T) are as follows:

$$\text{FLOPs}_{\text{KDA}}(T; C, d_h) = 6Td_h^2 + 3TCd_h + TC^2. \tag{13}$$

For full (global) attention, the dominant term per head is

$$\text{FLOPs}_{\text{Attn}}(T; d_h) \ = \ 2T^2 d_h. \tag{14}$$

Inference strategy and cost   The inference strategy in Kimi Linear employs a hybrid approach to optimize both computational and I/O efficiency. During the prefill phase, the model utilizes a FLOP-intensive chunk kernel (see § 3.1), while switching to the more efficient recurrent kernel (Eq. 2) for autoregressive generation. A key advantage of the Linear KDA is its ability to maintain a fixed-sized state ($d_k \times d_v$ per head, with $d_k = d_v = 128$) regardless of sequence length. For our hybrid model, as sequence length increases, the I/O-bounded decoding time approaches a maximum hybrid efficiency ratio of 3:1 compared to full attention. This trend is reflected in Fig. 7b, where Kimi Linear achieves a $2.3\times$ speedup at a 1M token context. Additionally, by eliminating the need for a large, linear-scaling KV cache, Kimi Linear is able to reallocate memory resources to support larger batch sizes, enhancing overall throughput. In long-context scenarios (up to 1M tokens), this memory efficiency results in a theoretical decoding speedup of up to $6.3\times$ (see Fig. 1b).

15

Table 7: An overview of different attention mechanisms through the lens of state updating rules and their learning objective under the TTT framework [89]. We ignore all normalizer terms and activation/kernel functions for brevity.

	Objective $\mathcal{L}$	Update rule $\mathbf{S}_t = \mathbf{S}_{t-1} - \nabla_{\mathbf{S}_{t-1}}\mathcal{L}$
LA [47]	$-\langle \mathbf{S}_{t-1}^\top \boldsymbol{k}_t, \boldsymbol{v}_t \rangle$	$\mathbf{S}_t = \mathbf{S}_{t-1} + \boldsymbol{k}_t \boldsymbol{v}_t^\top$
RetNet [91]	$-\beta_t \langle \mathbf{S}_{t-1}^\top \boldsymbol{k}_t, \boldsymbol{v}_t \rangle + \frac{1}{2}\left\|\sqrt{1-\alpha}\,\mathbf{S}_{t-1}\right\|_F^2$	$\mathbf{S}_t = \alpha \mathbf{S}_{t-1} + \beta_t \boldsymbol{k}_t \boldsymbol{v}_t^\top$
Mamba2 [16]	$-\beta_t \langle \mathbf{S}_{t-1}^\top \boldsymbol{k}_t, \boldsymbol{v}_t \rangle + \frac{1}{2}\left\|\sqrt{1-\alpha_t}\mathbf{S}_{t-1}\right\|_F^2$	$\mathbf{S}_t = \alpha_t \mathbf{S}_{t-1} + \beta_t \boldsymbol{k}_t \boldsymbol{v}_t^\top$
GLA [113]	$-\langle \mathbf{S}_{t-1}^\top \boldsymbol{k}_t, \boldsymbol{v}_t \rangle + \frac{1}{2}\left\|\sqrt{\mathrm{Diag}\,(\mathbf{1}-\boldsymbol{\alpha}_t)}\mathbf{S}_{t-1}\right\|_F^2$	$\mathbf{S}_t = \mathrm{Diag}(\boldsymbol{\alpha}_t)\mathbf{S}_{t-1} + \boldsymbol{k}_t \boldsymbol{v}_t^\top$
HGRN2 [76]	$-\langle \mathbf{S}_{t-1}^\top(\mathbf{1}-\boldsymbol{\alpha}_t), \boldsymbol{v}_t \rangle + \frac{1}{2}\left\|\sqrt{\mathrm{Diag}\,(\mathbf{1}-\boldsymbol{\alpha}_t)}\mathbf{S}_{t-1}\right\|_F^2$	$\mathbf{S}_t = \mathrm{Diag}(\boldsymbol{\alpha}_t)\mathbf{S}_{t-1} + (\mathbf{1}-\boldsymbol{\alpha}_t)\boldsymbol{v}_t^\top$
Longhorn [58]	$\frac{1}{2}\left\|\mathbf{S}_{t-1}^\top \boldsymbol{k}_t - \boldsymbol{v}_t\right\|_{\mathrm{Diag}(\boldsymbol{\beta}_t)}^2$	$\mathbf{S}_t = \left(\mathbf{I} - \frac{\beta_t}{1+\beta_t \boldsymbol{k}_t^\top \boldsymbol{k}_t}\boldsymbol{k}_t \boldsymbol{k}_t^\intercal\right)\mathbf{S}_{t-1} + \beta_t \boldsymbol{k}_t \boldsymbol{v}_t^\intercal$
Comba [39]	$\frac{\beta_t}{2}\left\|\mathbf{S}_{t-1}^\top \boldsymbol{k}_t - \boldsymbol{v}_t\right\|^2 + \frac{1}{2}\left\|\sqrt{1-\alpha_t}\mathbf{S}_{t-1}\right\|_F^2$	$\mathbf{S}_t = \left(\alpha_t - \beta_t \boldsymbol{k}_t \hat{\boldsymbol{k}}_t^\top\right)\mathbf{S}_{t-1} + \beta_t \boldsymbol{k}_t \boldsymbol{v}_t^\top$
RWKV7 [70]	$\frac{\beta_t}{2}\left\|\mathbf{S}_{t-1}^\top \boldsymbol{k}_t - \frac{1}{\beta_t}\boldsymbol{v}_t\right\|^2 + \frac{1}{2}\left\|\sqrt{\mathrm{Diag}\,(\mathbf{1}-\boldsymbol{\alpha}_t)}\mathbf{S}_{t-1}\right\|_F^2$	$\mathbf{S}_t = \left(\mathrm{Diag}\,(\boldsymbol{\alpha}_t) - \beta_t \hat{\boldsymbol{k}}_t \hat{\boldsymbol{k}}_t^\top\right)\mathbf{S}_{t-1} + \tilde{\boldsymbol{k}}_t \boldsymbol{v}_t^\top$
GDN [110]	$\frac{\beta_t}{2}\left\|\tilde{\mathbf{S}}_{t-1}^\top \boldsymbol{k}_t - \boldsymbol{v}_t\right\|^2$	$\mathbf{S}_t = \left(\mathbf{I} - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top\right)\alpha_t \mathbf{S}_{t-1} + \beta_t \boldsymbol{k}_t \boldsymbol{v}_t^\top$
KDA (ours)	$\frac{\beta_t}{2}\left\|\tilde{\mathbf{S}}_{t-1}^\top \boldsymbol{k}_t - \boldsymbol{v}_t\right\|^2$	$\mathbf{S}_t = \left(\mathbf{I} - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top\right)\mathrm{Diag}\,(\boldsymbol{\alpha}_t)\mathbf{S}_{t-1} + \beta_t \boldsymbol{k}_t \boldsymbol{v}_t^\top$

For GDN and KDA, the update can be viewed as performing an Stochastic Gradient Descent(SGD) process on the decayed state $\tilde{\mathbf{S}}$, that is, $\mathbf{S}_t = \tilde{\mathbf{S}}_{t-1} - \nabla_{\tilde{\mathbf{S}}_{t-1}}\mathcal{L}$, where $\tilde{\mathbf{S}}_{t-1}$ is decayed by scaler or fine-grained gate.

7 Related Works

7.1 Efficient Subquadratic Attention

The quadratic time complexity of the standard self-attention mechanism [98] remains a fundamental bottleneck for processing long contexts in Transformer-based models. This limitation has become increasingly critical as large language models (LLMs) are now expected to handle million-token sequences for tasks such as agentic tool use and repository-level code analysis [19, 49]. To overcome this challenge, a substantial body of research has explored more efficient attention mechanisms [90, 88], which can broadly be categorized into two main directions: (1) linear attention, and (2) sparse attention.

Linear Attention reformulates the quadratic attention map into kernelized feature interactions, replacing the softmax with a positive feature map so that attention can be computed through two associative matrix products [47]. This eliminates the explicit $\mathcal{O}(T^2)$ similarity matrix and enables linear-time computation with respect to sequence length. Subsequent work strengthens the vanilla linear attention significantly through more refined memory control, shifting from data-independent "decay" [91, 77] to more adaptive, data-dependent mechanisms [29, 92], and refining the decay granularity from coarse headwise [16] to precise, channel-wise decay. GLA generalizes these approaches with diagonal, channel-wise gates that balance expressiveness and efficiency while retaining chunk-wise parallelism [112, 113]. Table 7 summarizes the corresponding update rules. Collectively, these methods cast attention as a compact recurrent memory updated with parallel prefix-scan operators and fused matrix multiplies, aligning well with modern accelerators [41].

A complementary view connects linear attention to *fast-weight* memory [83]: the state is a low-capacity associative table updated online by Hebbian-like rules [68], while slow weights amortize when to store, update, or forget [67].

In Table 7, we provide a summary of the existing efficient token mixing methods, comparing them from the perspectives of state update mechanisms and optimization objectives.

From this perspective, gating and decay serve as learnable criteria that mitigate interference and stabilize optimization [89]. Despite these advances, linear attention still lags full attention on exact copying and fine-grained selection in extreme long-context retrieval. This motivates hybrid designs (interleaving linear and full attention) and more structured updates. In particular, the gated delta rule used by GDN/KDA introduces rank-1 corrective updates to the fast-weight state, improving targeted retention while remaining parallelizable at the operator level [111].

Linear Attention with Gating Mechanism The vanilla Linear Attention [47] is known to lack the selection mechanism inherent in softmax attention [98], falling short in expressiveness. To address this, Gated Linear Attention models have emerged as memory-efficient and parallelizable alternatives [112, 113, 29]. Instead of storing an ever-expanding KV cache, these models employ a fixed-size matrix-valued state and learnable gates to selectively retain

and forget information. This design achieves expressive power comparable to softmax attention [64, 124, 63] while maintaining constant time and memory complexity during inference time. The general recurrent formulation of such models for memory update $\mathbf{S}_t \in \mathbb{R}^{d_k \times d_v}$ can be expressed as:

$$\mathbf{S}_t = \mathbf{A}_t \mathbf{S}_{t-1} + \boldsymbol{k}_t \boldsymbol{v}_t^\top, \quad \boldsymbol{o}_t = \mathbf{S}_t^\top \boldsymbol{q}_t. \tag{15}$$

The primary distinction among various gated linear attention mechanisms lies in the parameterization of the forget gate $\mathbf{A}_t$, as summarized in Table 7. For instance, RetNet [91] uses a data-independent scalar decay α, and Mamba2 [16] employs a data-dependent scalar α_t. Specifically, GLA [113] utilized a diagonalized fine-grained matrix $\mathrm{Diag}(\boldsymbol{\alpha}_t) \in \mathbb{R}^{d_k \times d_k}$, offering an effective trade-off between efficiency and performance. Other variants are displayed in Table 7.

Sparse Attention   A separate body of work reduces the quadratic complexity of standard attention by exploiting its inherent sparsity, approximating the full attention score by performing the computation on a strategically selected subset of tokens. The central challenge lies in identifying this subset effectively without degrading model performance. Early methods often utilized efficient, training-free static patterns, such as sliding and dilated windows [20, 30, 106], or fixed patterns [119, 34], but their rigid structure often compromises model accuracy. More advanced methods determine the important positions based on the context, such as clustering [50, 105] and lightweight routing mechanisms [25, 72, 2, 9], but this dynamic selection process introduces a computational overhead that can prevent them from achieving their full theoretical speedup without dedicated kernel acceleration [21]. Some models further introduce training-free sparsification during the inference stage [106, 108].

Recent approaches to sparse attention have begun to prioritize hardware co-design, as exemplified by NSA [118, 95] and MoBA [62], which both move from token-level to chunk-level selection. In NSA, each query dynamically selects chunks based on scores produced by an MLP. The method's efficiency relies on its use of Grouped-Query Attention (GQA) [97] with a large head count (typically a multiple of 16), a configuration specifically designed to accelerate computation through highly parallelized tensor–matrix multiplications. Similarly, MoBA performs top-k chunk selection, but leverages log-sum-exp (LSE) scores computed efficiently via flash-attention kernels [17]. In contrast to NSA and MoBA, the recently proposed DeepSeek-V3.2-Exp Attention (DSA) [18] revives token-level sparsity, maintaining efficiency through a learnable full-attention indexer implemented with low-precision fp8 and a small head dimension for token selection.

Discussion   Linear attention and sparse attention represent two distinct pathways toward efficient long-context modeling. Sparse attention tends to retrieve fine-grained historical information more effectively, but this advantage comes at the cost of storing the entire KV cache for token selection, making it less efficient than linear attention models that maintain a constant state. Moreover, sparse attention performs only information selection, and its theoretical expressive upper bound remains that of full attention. In contrast, linear attention, grounded in the principle of "compression as intelligence", enables generalization with a fixed-size state and, when combined with the Delta learning rule, can achieve theoretically stronger expressive capacity. Although linear attentions have traditionally been criticized for weak retrieval ability, this limitation can be mitigated through state expansion [23, 33, 116, 38] or related techniques. Nevertheless, despite these advantages, linear attention remains limited by current hardware implementations and the absence of optimized inference infrastructure. Our work overcomes these limitations with KDA-H, a powerful model integrated with vLLM for efficient inference. Our proposed KDA delivers competitive performance compared to the full-attention baseline (Table 3) and achieves over a $2\times$ decoding speedup at the one-million-token context (Figure 7b). Despite their distinct approaches to efficient long-context modeling, linear attention and sparse attention are not mutually exclusive. Future work could explore hybrid models that integrate the strengths of both, leveraging the compression and generalization capabilities of linear attention with the fine-grained retrieval advantages of sparse attention to further enhance model performance and efficiency.

7.2   Hybrid Model

Despite efficiency, pure Linear Attention still struggle with precise memory retrieval and exact copying [44, 103] This deficiency hinders their adoption in industrial-scale LLMs where robust long-context recall (e.g., beyond 1M tokens) and reliable tool-use over extensive code repositories are critical [49]. Recent work shows that Linear Attention and full attention can effectively complement each other, leading to various hybrid designs.

Intra-layer hybrid   One category of hybrid architectures is the intra-layer hybrid, which adaptively fuses the outputs of different mechanisms within each layer. A common implementation fuses outputs from heterogeneous heads within each layer, such as combining standard attention with state space models (SSMs) [22, 55]. In contrast, sequence-level approaches apply distinct mechanisms to different parts of the input. For example, some use linear attention for past context and SWA for recent tokens [122, 53, 66], while NHA [24] compresses the history with GSA [123] and combines it with local sliding window context to emulate a standard attention operation.

Inter-layer Hybrid   A key drawback of the intra-layer hybrid is the increased system complexity and inference overhead. The heterogeneous mechanisms require separate computational paths, complicating optimizations like distributed parallelism. To mitigate this challenge, inter-layer hybrids have become a more widely adopted and practical strategy in LLMs [65, 56, 96]. This approach involves stacking distinct layer types, such as full attention and a linear alternative, in a predefined ratio. Building on this paradigm, we implement a simple yet effective strategy: interleaving linear and full attention layers at a fixed 3:1 ratio (see § 5.2 for ablations). This regular, repeating structure simplifies KV cache management and integrates seamlessly with standard optimizations. For the linear component of our hybrid, we deviate from the common practice of using Mamba2 [16]. Instead, we employ KDA, as we found it yields superior overall performance, particularly in retrieval and copying abilities.

Discussion   Recent work indicates that hybrid models can be sensitive to adjustments in the RoPE base frequency, a vulnerability that complicates context window extension [125]. This sensitivity can hinder the model's ability to extrapolate to longer sequences. To address this challenge, recent models have trended towards solutions that incorporate No Position Embeddings (NoPE). Falcon-H [125], for example, uses an unconventionally high base frequency (e.g., $b \approx 10^{11}$) to push its positional encoding to a near-NoPE state. Architecturally, SwanGPT [75] interleaves RoPE-based layers with NoPE-based full attention layers. Aligning with this direction, we found that hybridizing our KDA layers with NoPE full attention is also a highly effective strategy, facilitating straightforward context window extension.

Conclusion

We introduce Kimi Linear, a hybrid linear attention architecture designed to meet the efficiency demands of agentic intelligence and test-time scaling without sacrificing quality. Central to Kimi Linear is Kimi Delta Attention (KDA), an advanced linear attention module with a channel-wise gating mechanism that enhances memory control and enables RNN-style models in hybrid architectures. By interleaving KDA with global attention in a 3:1 ratio, Kimi Linear reduces memory usage by up to 75%, while achieving up to $6.3\times$ higher decoding throughput and outperforming full-attention baselines. Our approach provides a scalable, efficient solution for large language models, with open-source KDA kernels and pre-trained checkpoints facilitating further research.

References

[1] Sandhini Agarwal et al. "gpt-oss-120b & gpt-oss-20b model card". In: *arXiv preprint arXiv:2508.10925* (2025).

[2] Joshua Ainslie et al. "Colt5: Faster long-range transformers with conditional computation". In: *arXiv preprint arXiv:2303.09752* (2023).

[3] Zeyuan Allen-Zhu. "Physics of Language Models: Part 4.1, Architecture Design and the Magic of Canon Layers". In: *SSRN Electronic Journal* (May 2025). Available at SSRN: https://ssrn.com/abstract=5240330 or http://dx.doi.org/10.2139/ssrn.5240330. DOI: 10.2139/ssrn.5240330.

[4] Simran Arora et al. "Simple linear attention language models balance the recall-throughput tradeoff". In: *Forty-first International Conference on Machine Learning*. 2024. URL: https://openreview.net/forum?id=e93ffDcpH3.

[5] Simran Arora et al. *Zoology: Measuring and Improving Recall in Efficient Language Models*. 2023. arXiv: 2312.04927 [cs.CL].

[6] Yushi Bai et al. "Longbench v2: Towards deeper understanding and reasoning on realistic long-context multi-tasks". In: *arXiv preprint arXiv:2412.15204* (2024).

[7] Federico Barbero et al. "Round and Round We Go! What makes Rotary Positional Encodings useful?" In: *Proceedings of ICLR*. 2025. URL: https://openreview.net/forum?id=GtvuNrk58a.

[8] Ali Behrouz et al. "Atlas: Learning to optimally memorize the context at test time". In: *arXiv preprint arXiv:2505.23735* (2025).

[9] Amanda Bertsch et al. "Unlimiformer: Long-range transformers with unlimited length input". In: *Advances in NIPS* 36 (2023), pp. 35522–35543.

[10] Stella Biderman et al. "Lessons from the trenches on reproducible evaluation of language models". In: *arXiv preprint arXiv:2405.14782* (2024).

[11] Christian Bischof and Charles Van Loan. "The WY Representation for Products of Householder Matrices". In: *SIAM Journal on Scientific and Statistical Computing* (1987), s2–s13. URL: https://doi.org/10.1137/0908009.

[12] Aaron Blakeman et al. "Nemotron-h: A family of accurate and efficient hybrid mamba-transformer models". In: *arXiv preprint arXiv:2504.03624* (2025).

[13] Egor Bogomolov et al. "Long code arena: a set of benchmarks for long-context code models". In: *arXiv preprint arXiv:2406.11612* (2024).

[14] Peter Clark et al. "Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge". In: *arXiv:1803.05457v1* (2018).

[15] Ganqu Cui et al. "The entropy mechanism of reinforcement learning for reasoning language models". In: *arXiv preprint arXiv:2505.22617* (2025).

[16] Tri Dao and Albert Gu. "Transformers are SSMs: Generalized Models and Efficient Algorithms Through Structured State Space Duality". In: *CoRR* abs/2405.21060 (2024). DOI: 10.48550/ARXIV.2405.21060. arXiv: 2405.21060. URL: https://doi.org/10.48550/arXiv.2405.21060.

[17] Tri Dao et al. "FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness". In: *Advances in NIPS*. 2022, pp. 16344–16359. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/67d57c32e20fd0a7a302cb81d36e40d5-Paper-Conference.pdf.

[18] DeepSeek-AI. *DeepSeek-V3.2-Exp: Boosting Long-Context Efficiency with DeepSeek Sparse Attention*. 2025.

[19] DeepSeek-AI et al. *DeepSeek-V3 Technical Report*. 2025. arXiv: 2412.19437 [cs.CL]. URL: https://arxiv.org/abs/2412.19437.

[20] Jiayu Ding et al. *LongNet: Scaling Transformers to 1,000,000,000 Tokens*. 2023. arXiv: 2307.02486 [cs.CL]. URL: https://arxiv.org/abs/2307.02486.

[21] Juechu Dong et al. *Flex Attention: A Programming Model for Generating Optimized Attention Kernels*. 2024. arXiv: 2412.05496 [cs.LG]. URL: https://arxiv.org/abs/2412.05496.

[22] Xin Dong et al. *Hymba: A Hybrid-head Architecture for Small Language Models*. 2024. arXiv: 2411.13676 [cs.CL]. URL: https://arxiv.org/abs/2411.13676.

[23] Jusen Du et al. "Mom: Linear sequence modeling with mixture-of-memories". In: *arXiv preprint arXiv:2502.13685* (2025).

[24] Jusen Du et al. "Native Hybrid Attention for Efficient Sequence Modeling". In: *arXiv preprint arXiv:2510.07019* (2025).

[25] Tianyu Fu et al. "Moa: Mixture of sparse attention for automatic large language model compression". In: *arXiv preprint arXiv:2406.14909* (2024).

[26] Aryo Pradipta Gema et al. "Are we done with mmlu?" In: *arXiv preprint arXiv:2406.04127* (2024).

[27] Riccardo Grazzi et al. "Unlocking State-Tracking in Linear RNNs Through Negative Eigenvalues". In: *Proceedings of ICLR*. 2025. URL: https://openreview.net/forum?id=UvTo3tVBk2.

[28] Joseph F. Grcar. "How ordinary elimination became Gaussian elimination". In: *Historia Mathematica* 38.2 (May 2011), pp. 163–218. ISSN: 0315-0860. DOI: 10.1016/j.hm.2010.06.003. URL: http://dx.doi.org/10.1016/j.hm.2010.06.003.

[29] Albert Gu and Tri Dao. *Mamba: Linear-Time Sequence Modeling with Selective State Spaces*. 2023. arXiv: 2312.00752 [cs.LG].

[30] Xiangming Gu et al. *When Attention Sink Emerges in Language Models: An Empirical View*. 2025. arXiv: 2410.10781 [cs.CL]. URL: https://arxiv.org/abs/2410.10781.

[31] Yuxian Gu et al. *Jet-Nemotron: Efficient Language Model with Post Neural Architecture Search*. 2025. arXiv: 2508.15884 [cs.CL]. URL: https://arxiv.org/abs/2508.15884.

[32] Daya Guo et al. "DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning". In: *Nature* 645.8081 (2025), pp. 633–638.

[33] Han Guo et al. "Log-linear attention". In: *arXiv preprint arXiv:2506.04761* (2025).

[34] Qipeng Guo et al. "Star-transformer". In: *arXiv preprint arXiv:1902.09113* (2019).

[35] Dan Hendrycks et al. *Measuring Massive Multitask Language Understanding*. 2021. arXiv: 2009.03300 [cs.CY]. URL: https://arxiv.org/abs/2009.03300.

[36] Jordan Hoffmann et al. *Training Compute-Optimal Large Language Models*. 2022. arXiv: 2203.15556 [cs.CL]. URL: https://arxiv.org/abs/2203.15556.

[37] Cheng-Ping Hsieh et al. "RULER: What's the Real Context Size of Your Long-Context Language Models?" In: *arXiv preprint arXiv:2404.06654* (2024).

[38] Jiaxi Hu et al. "Attractor memory for long-term time series forecasting: A chaos perspective". In: *Advances in NIPS* 37 (2024), pp. 20786–20818.

[39] Jiaxi Hu et al. "Comba: Improving Nonlinear RNNs with Closed-loop Control". In: *arXiv preprint arXiv:2506.02475* (2025).

[40] Ermo Hua et al. "Fourier Position Embedding: Enhancing Attention's Periodic Extension for Length Generalization". In: *arXiv preprint arXiv:2412.17739* (2024).

[41] Weizhe Hua et al. "Transformer Quality in Linear Time". In: *Proceedings of ICML*. Ed. by Kamalika Chaudhuri et al. PMLR, 2022, pp. 9099–9117. URL: https://proceedings.mlr.press/v162/hua22a.html.

[42] Yuzhen Huang et al. "C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models". In: *Advances in NIPS* 36 (2023), pp. 62991–63010.

[43] Naman Jain et al. "Livecodebench: Holistic and contamination free evaluation of large language models for code". In: *arXiv preprint arXiv:2403.07974* (2024).

[44] Samy Jelassi et al. *Repeat After Me: Transformers are Better than State Space Models at Copying*. 2024. arXiv: 2402.01032 [cs.LG].

[45] Thierry Joffrain et al. "Accumulating Householder transformations, revisited". In: (2006), pp. 169–179. URL: https://doi.org/10.1145/1141885.1141886.

[46] Mandar Joshi et al. "Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension". In: *arXiv preprint arXiv:1705.03551* (2017).

[47] Angelos Katharopoulos et al. "Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention". In: *Proceedings of ICML*. Ed. by Hal Daumé III and Aarti Singh. PMLR, 2020, pp. 5156–5165. URL: https://proceedings.mlr.press/v119/katharopoulos20a.html.

[48] Amirhossein Kazemnejad et al. "The impact of positional encoding on length generalization in transformers". In: *Advances in NIPS* 36 (2023), pp. 24892–24928.

[49] Team Kimi et al. "Kimi k2: Open agentic intelligence". In: *arXiv preprint arXiv:2507.20534* (2025).

[50] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. "Reformer: The efficient transformer". In: *arXiv preprint arXiv:2001.04451* (2020).

[51] Satyapriya Krishna et al. "Fact, fetch, and reason: A unified evaluation of retrieval-augmented generation". In: *arXiv preprint arXiv:2409.12941* (2024).

[52] Hanyu Lai et al. "A Survey of Post-Training Scaling in Large Language Models". In: *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2025, pp. 2771–2791.

[53] Disen Lan et al. "Liger: Linearizing Large Language Models to Gated Recurrent Structures". In: *arXiv preprint arXiv:2503.01496* (2025).

[54] Haonan Li et al. "CMMLU: Measuring massive multitask language understanding in Chinese". In: *Findings of the Association for Computational Linguistics: ACL 2024*. Ed. by Lun-Wei Ku, Andre Martins, and Vivek Srikumar. Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 11260–11285. DOI: `10.18653/v1/2024.findings-acl.671`. URL: `https://aclanthology.org/2024.findings-acl.671/`.

[55] Yixing Li et al. "Transmamba: Flexibly switching between transformer and mamba". In: *arXiv preprint arXiv:2503.24067* (2025).

[56] Opher Lieber et al. *Jamba: A Hybrid Transformer-Mamba Language Model*. 2024. arXiv: `2403.19887 [cs.CL]`.

[57] Zhixuan Lin et al. "Forgetting transformer: Softmax attention with a forget gate". In: *arXiv preprint arXiv:2503.02130* (2025).

[58] Bo Liu et al. "Longhorn: State Space Models are Amortized Online Learners". In: *ArXiv* abs/2407.14207 (2024). URL: `https://api.semanticscholar.org/CorpusID:271310065`.

[59] Jiawei Liu et al. "Is Your Code Generated by ChatGPT Really Correct? Rigorous Evaluation of Large Language Models for Code Generation". In: *Thirty-seventh Conference on NIPS*. 2023. URL: `https://openreview.net/forum?id=1qvx610Cu7`.

[60] Jiawei Liu et al. "Repoqa: Evaluating long context code understanding". In: *arXiv preprint arXiv:2406.06025* (2024).

[61] Jingyuan Liu et al. *Muon is Scalable for LLM Training*. 2025. arXiv: `2502.16982 [cs.LG]`. URL: `https://arxiv.org/abs/2502.16982`.

[62] Enzhe Lu et al. *MoBA: Mixture of Block Attention for Long-Context LLMs*. 2025. arXiv: `2502.13189 [cs.LG]`. URL: `https://arxiv.org/abs/2502.13189`.

[63] William Merrill, Jackson Petty, and Ashish Sabharwal. "The illusion of state in state-space models". In: *arXiv preprint arXiv:2404.08819* (2024).

[64] William Merrill and Ashish Sabharwal. "The Parallelism Tradeoff: Limitations of Log-Precision Transformers". In: *Transactions of the Association for Computational Linguistics* 11 (2023), pp. 531–545. DOI: `10.1162/tacl_a_00562`. URL: `https://aclanthology.org/2023.tacl-1.31/`.

[65] MiniMax et al. *MiniMax-01: Scaling Foundation Models with Lightning Attention*. 2025. arXiv: `2501.08313 [cs.CL]`.

[66] Tsendsuren Munkhdalai, Manaal Faruqui, and Siddharth Gopal. *Leave No Context Behind: Efficient Infinite Context Transformers with Infini-attention*. 2024. arXiv: `2404.07143 [cs.CL]`.

[67] Tsendsuren Munkhdalai and Adam Trischler. *Metalearning with Hebbian Fast Weights*. 2018. arXiv: `1807.05076 [cs.NE]`. URL: `https://arxiv.org/abs/1807.05076`.

[68] Tsendsuren Munkhdalai et al. "Metalearned Neural Memory". In: *ArXiv* abs/1907.09720 (2019). URL: `https://api.semanticscholar.org/CorpusID:198179407`.

[69] Long Ouyang et al. "Training language models to follow instructions with human feedback". In: *Advances in NIPS* 35 (2022), pp. 27730–27744.

[70] Bo Peng et al. *RWKV-7 "Goose" with Expressive Dynamic State Evolution*. 2025. arXiv: `2503.14456 [cs.CL]`.

[71] Bowen Peng et al. "Yarn: Efficient context window extension of large language models". In: *arXiv preprint arXiv:2309.00071* (2023).

[72] Piotr Piękos, Róbert Csordás, and Jürgen Schmidhuber. "Mixture of Sparse Attention: Content-Based Learnable Sparse Attention via Expert-Choice Routing". In: *arXiv preprint arXiv:2505.00315* (2025).

[73] Aske Plaat et al. "Reasoning with large language models, a survey". In: *CoRR* (2024).

[74] Ofir Press, Noah Smith, and Mike Lewis. "Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation". In: *Proceedings of ICLR*. 2022. URL: `https://openreview.net/forum?id=R8sQPpGCv0`.

[75] Krishna C. Puvvada et al. *SWAN-GPT: An Efficient and Scalable Approach for Long-Context Language Modeling*. 2025. arXiv: `2504.08719 [cs.CL]`.

[76] Zhen Qin et al. *HGRN2: Gated Linear RNNs with State Expansion*. 2024. arXiv: `2404.07904 [cs.CL]`.

[77] Zhen Qin et al. *TransNormerLLM: A Faster and Better Large Language Model with Improved TransNormer*. 2024. arXiv: `2307.14995 [cs.CL]`.

[78] Zihan Qiu et al. *Gated Attention for Large Language Models: Non-linearity, Sparsity, and Attention-Sink-Free*. 2025. arXiv: `2505.06708 [cs.CL]`.

[79] Xiaoye Qu et al. "A survey of efficient reasoning for large reasoning models: Language, multimodality, and beyond". In: *arXiv preprint arXiv:2503.21614* (2025).

[80] Qwen Team. *Qwen3-Next: Towards Ultimate Training & Inference Efficiency*. Accessed: 2025-10-27. Sept. 2025.

[81] David Rein et al. "Gpqa: A graduate-level google-proof q&a benchmark". In: *First Conference on Language Modeling*. 2024.

[82] Keisuke Sakaguchi et al. *WinoGrande: An Adversarial Winograd Schema Challenge at Scale*. 2019. arXiv: 1907.10641 [cs.CL]. URL: https://arxiv.org/abs/1907.10641.

[83] Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. "Linear Transformers Are Secretly Fast Weight Programmers". In: *Proceedings of ICML*. Ed. by Marina Meila and Tong Zhang. PMLR, 2021, pp. 9355–9366. URL: https://proceedings.mlr.press/v139/schlag21a.html.

[84] Imanol Schlag, Tsendsuren Munkhdalai, and Jürgen Schmidhuber. *Learning Associative Inference Using Fast Weight Memory*. 2021. arXiv: 2011.07831 [cs.LG]. URL: https://arxiv.org/abs/2011.07831.

[85] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. *Self-Attention with Relative Position Representations*. 2018. arXiv: 1803.02155 [cs.CL].

[86] Jianlin Su. *Linear Attention: A Brief History of Imitation, Innovation, and Feedback*. June 2025. URL: https://kexue.fm/archives/11033.

[87] Jianlin Su et al. "Roformer: Enhanced transformer with rotary position embedding". In: *Neurocomputing* 568 (2024), p. 127063.

[88] Weigao Sun et al. *Speed Always Wins: A Survey on Efficient Architectures for Large Language Models*. 2025. arXiv: 2508.09834 [cs.CL]. URL: https://arxiv.org/abs/2508.09834.

[89] Yu Sun et al. "Learning to (Learn at Test Time): RNNs with Expressive Hidden States". In: *ArXiv* abs/2407.04620 (2024). URL: https://api.semanticscholar.org/CorpusID:271039606.

[90] Yutao Sun et al. "Efficient attention mechanisms for large language models: A survey". In: *arXiv preprint arXiv:2507.19595* (2025).

[91] Yutao Sun et al. *Retentive Network: A Successor to Transformer for Large Language Models*. 2023. arXiv: 2307.08621 [cs.CL].

[92] Yutao Sun et al. *You Only Cache Once: Decoder-Decoder Architectures for Language Models*. 2024. arXiv: 2405.05254 [cs.CL]. URL: https://arxiv.org/abs/2405.05254.

[93] Mirac Suzgun et al. "Challenging big-bench tasks and whether chain-of-thought can solve them". In: *arXiv preprint arXiv:2210.09261* (2022).

[94] Kimi Team et al. *Kimi k1.5: Scaling Reinforcement Learning with LLMs*. 2025. arXiv: 2501.12599 [cs.AI]. URL: https://arxiv.org/abs/2501.12599.

[95] MiniCPM Team et al. *MiniCPM4: Ultra-Efficient LLMs on End Devices*. 2025. arXiv: 2506.07900 [cs.CL]. URL: https://arxiv.org/abs/2506.07900.

[96] Tencent Hunyuan Team et al. "Hunyuan-turbos: Advancing large language models through mamba-transformer synergy and adaptive chain-of-thought". In: *arXiv preprint arXiv:2505.15431* (2025).

[97] Hugo Touvron et al. *LLaMA: Open and Efficient Foundation Language Models*. 2023. arXiv: 2302.13971 [cs.CL].

[98] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in NIPS*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[99] Roger Waleffe et al. *An Empirical Study of Mamba-based Language Models*. 2024. arXiv: 2406.07887 [cs.LG]. URL: https://arxiv.org/abs/2406.07887.

[100] Sinong Wang et al. *Linformer: Self-Attention with Linear Complexity*. 2020. arXiv: 2006.04768 [cs.LG].

[101] Yaoyu Wang. *Understanding DeltaNet from the Perspective of Inference Frameworks*. May 2025. URL: https://yywangcs.notion.site/DeltaNet-1fefc9f5d80580a496f8eb406a496f09.

[102] Yubo Wang et al. "Mmlu-pro: A more robust and challenging multi-task language understanding benchmark". In: *Advances in NIPS* 37 (2024), pp. 95266–95290.

[103] Kaiyue Wen, Xingyu Dang, and Kaifeng Lyu. "Rnns are not transformers (yet): The key bottleneck on in-context retrieval". In: *arXiv preprint arXiv:2402.18510* (2024).

[104] Colin White et al. "Livebench: A challenging, contamination-free llm benchmark". In: *arXiv preprint arXiv:2406.19314* 4 (2024).

[105] Yuhuai Wu et al. "Memorizing transformers". In: *arXiv preprint arXiv:2203.08913* (2022).

[106] Guangxuan Xiao et al. "Efficient streaming language models with attention sinks". In: *arXiv preprint arXiv:2309.17453* (2023).

[107] Wenhan Xiong et al. *Effective Long-Context Scaling of Foundation Models*. 2023. arXiv: 2309.16039 [cs.CL]. URL: https://arxiv.org/abs/2309.16039.

[108] Ruyi Xu et al. "Xattention: Block sparse attention with antidiagonal scoring". In: *arXiv preprint arXiv:2503.16428* (2025).

[109] Bowen Yang et al. *Rope to Nope and Back Again: A New Hybrid Attention Strategy*. 2025. arXiv: 2501.18795 [cs.CL]. URL: https://arxiv.org/abs/2501.18795.

[110] Songlin Yang, Jan Kautz, and Ali Hatamizadeh. "Gated Delta Networks: Improving Mamba2 with Delta Rule". In: *Proceedings of ICLR*. 2025. URL: https://openreview.net/forum?id=r8H7xhYPwz.

[111] Songlin Yang and Bailin Wang. "Parallelizing Linear Transformers with the Delta Rule over Sequence Length". In: *ArXiv* abs/2406.06484 (2024). URL: https://api.semanticscholar.org/CorpusID:270371554.

[112] Songlin Yang and Yu Zhang. *FLA: A Triton-Based Library for Hardware-Efficient Implementations of Linear Attention Mechanism*. 2024. URL: https://github.com/fla-org/flash-linear-attention.

[113] Songlin Yang et al. "Gated Linear Attention Transformers with Hardware-Efficient Training". In: *Proceedings of ICML*. PMLR, 2024.

[114] Songlin Yang et al. "PaTH Attention: Position Encoding via Accumulating Householder Transformations". In: *arXiv preprint arXiv:2505.16381* (2025).

[115] Feng Yao et al. *Your Efficient RL Framework Secretly Brings You Off-Policy RL Training*. Aug. 2025. URL: https://fengyao.notion.site/off-policy-rl.

[116] Morris Yau et al. "Sequential-Parallel Duality in Prefix Scannable Models". In: *arXiv preprint arXiv:2506.10918* (2025).

[117] Howard Yen et al. "HELMET: How to Evaluate Long-Context Language Models Effectively and Thoroughly". In: *International Conference on Learning Representations (ICLR)*. 2025.

[118] Jingyang Yuan et al. *Native Sparse Attention: Hardware-Aligned and Natively Trainable Sparse Attention*. 2025. arXiv: 2502.11089 [cs.CL]. URL: https://arxiv.org/abs/2502.11089.

[119] Manzil Zaheer et al. "Big bird: Transformers for longer sequences". In: *Advances in NIPS* 33 (2020), pp. 17283–17297.

[120] Rowan Zellers et al. "HellaSwag: Can a Machine Really Finish Your Sentence?" In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019.

[121] Biao Zhang and Rico Sennrich. "Root mean square layer normalization". In: *Advances in NIPS* 32 (2019).

[122] Michael Zhang et al. "Lolcats: On low-rank linearizing of large language models". In: *arXiv preprint arXiv:2410.10254* (2024).

[123] Yu Zhang et al. *Gated Slot Attention for Efficient Linear-Time Sequence Modeling*. 2024. arXiv: 2409.07146 [cs.CL].

[124] Shu Zhong et al. "Understanding Transformer from the Perspective of Associative Memory". In: *arXiv preprint arXiv:2505.19488* (2025).

[125] Jingwei Zuo et al. *Falcon-H1: A Family of Hybrid-Head Language Models Redefining Efficiency and Performance*. 2025. arXiv: 2507.22448 [cs.CL]. URL: https://arxiv.org/abs/2507.22448.

A   Contributions

The authors are listed in order of the significance of their contributions, with those in project leadership roles appearing last. The project is developed at Moonshot AI, with several external collaborators that are marked with #.

Yu Zhang[1]
Zongyu Lin
Xingcheng Yao
Jiaxi Hu[2]
Fanqing Meng
Chengyin Liu
Xin Men
Songlin Yang[#3]
Zhiyuan Li
Wentao Li
Enzhe Lu
Weizhou Liu
Yanru Chen
Weixin Xu
Longhui Yu
Yejie Wang
Yu Fan
Longguang Zhong
Enming Yuan
Dehao Zhang
Yizhi Zhang
T.Y. Liu
Haiming Wang
Shengjun Fang
Weiran He
Shaowei Liu
Yiwei Li
Jianlin Su
Jiezhong Qiu[4]
Bo Pang

Junjie Yan
Zhejun Jiang
Weixiao Huang
Bohong Yin
Jiacheng You
Chu Wei
Zhengtao Wang
Chao Hong
Yutian Chen
Guanduo Chen
Yucheng Wang
Huabin Zheng
Feng Wang
Yibo Liu
Mengnan Dong
Zheng Zhang
Siyuan Pan
Wenhao Wu
Yuhao Wu
Longyu Guan
Jiawen Tao
Guohong Fu[#1]
Xinran Xu
Yuzhi Wang
Guokun Lai
Yulun Du
Yuxin Wu
Xinyu Zhou
Zhilin Yang

[1] Soochow University, China

[2] The Hong Kong University of Science and Technology (Guangzhou)

[3] Massachusetts Institute of Technology

[4] Hangzhou Institute of Medicine, CAS

B   Derivations for Chunkwise Parallelism of KDA

We first recall the recurrent form of KDA:

$$\mathbf{S}_{[t]}^r = \underbrace{\left(\prod_{i=1}^{t}\left(\mathbf{I}-\beta_{[t]}^i\boldsymbol{k}_{[t]}^i\boldsymbol{k}_{[t]}^{i\top}\right)\operatorname{Diag}(\boldsymbol{\alpha}_{[t]}^i)\right)}_{:=\mathbf{P}_{[t]}^r}\cdot\mathbf{S}_{[t]}^0 + \underbrace{\sum_{i=1}^{r}\left(\prod_{j=i}^{t}\left(\mathbf{I}-\beta_{[t]}^j\boldsymbol{k}_{[t]}^j\boldsymbol{k}_{[t]}^{j\top}\right)\operatorname{Diag}(\boldsymbol{\alpha}_{[t]}^j)\right)\cdot\beta_{[t]}^i\boldsymbol{k}_{[t]}^i\boldsymbol{v}_{[t]}^{i\top}}_{:=\mathbf{H}_{[t]}^r}$$

$$= \mathbf{P}_{[t]}^r\cdot\mathbf{S}_{[t]}^r + \mathbf{H}_{[t]}^r$$

Our goal is to transform $\mathbf{P}_{[t]}^r$ and $\mathbf{H}_{[t]}^r$ into matrix forms suitable for parallel computation.

We show that $\mathbf{P}_{[t]}^r$, which involves the cumulative product of generalized Householder matrices, can be optimized using the classic WY representation.

Proposition 1. *The matrix $\mathbf{P}_{[t]}^r$ can be expressed as:*

$$\mathbf{P}_{[t]}^r = \operatorname{Diag}(\boldsymbol{\gamma}_{[t]}^r) - \sum_{i=1}^{r}\operatorname{Diag}(\boldsymbol{\gamma}_{[t]}^{i\to r})\boldsymbol{k}_{[t]}^i\boldsymbol{w}_{[t]}^{i\top} \tag{16}$$

where the auxiliary vector $\boldsymbol{w}_{[t]}^r \in \mathbb{R}^{d_k}$ is computed via the following recurrence relation:

$$\boldsymbol{w}_{[t]}^r = \beta_{[t]}^r\left(\operatorname{Diag}(\boldsymbol{\gamma}_{[t]}^r)\boldsymbol{k}_{[t]}^r - \sum_{i=1}^{r-1}\boldsymbol{w}_{[t]}^i\left(\boldsymbol{k}_{[t]}^{i\top}\operatorname{Diag}\left(\boldsymbol{\gamma}_{[t]}^{i\to r}\right)\boldsymbol{k}_{[t]}^r\right)\right) \tag{17}$$

Proof. We proceed with a proof by mathematical induction.

Inductive Step: Assume the proposition holds for $r-1$, i.e., $\mathbf{P}_{[t]}^{r-1} = \operatorname{Diag}(\boldsymbol{\gamma}_{[t]}^{r-1}) - \sum_{i=1}^{r-1}\operatorname{Diag}(\boldsymbol{\gamma}_{[t]}^{i\to r-1})\boldsymbol{k}_{[t]}^i\boldsymbol{w}_{[t]}^{i\top}$. We now derive:

$$\mathbf{P}_{[t]}^r = \left(\mathbf{I}-\beta_{[t]}^r\boldsymbol{k}_{[t]}^r\boldsymbol{k}_{[t]}^{r\top}\right)\operatorname{Diag}(\boldsymbol{\alpha}_{[t]}^r)\mathbf{P}_{[t]}^{r-1}$$

$$= \left(\mathbf{I}-\beta_{[t]}^r\boldsymbol{k}_{[t]}^r\boldsymbol{k}_{[t]}^{r\top}\right)\operatorname{Diag}(\boldsymbol{\alpha}_{[t]}^r)\left(\operatorname{Diag}(\boldsymbol{\gamma}_{[t]}^{r-1}) - \sum_{i=1}^{r-1}\operatorname{Diag}\left(\boldsymbol{\gamma}_{[t]}^{i\to r-1}\right)\boldsymbol{k}_{[t]}^i\boldsymbol{w}_{[t]}^{i\top}\right)$$

$$= \left(\mathbf{I}-\beta_{[t]}^r\boldsymbol{k}_{[t]}^r\boldsymbol{k}_{[t]}^{r\top}\right)\left(\operatorname{Diag}(\boldsymbol{\gamma}_{[t]}^r) - \sum_{i=1}^{r-1}\operatorname{Diag}\left(\boldsymbol{\gamma}_{[t]}^{i\to r}\right)\boldsymbol{k}_{[t]}^i\boldsymbol{w}_{[t]}^{i\top}\right)$$

$$= \operatorname{Diag}(\boldsymbol{\gamma}_{[t]}^r) - \sum_{i=1}^{r-1}\operatorname{Diag}\left(\boldsymbol{\gamma}_{[t]}^{i\to r}\right)\boldsymbol{k}_{[t]}^i\boldsymbol{w}_{[t]}^{i\top} - \beta_{[t]}^r\boldsymbol{k}_{[t]}^r\boldsymbol{k}_{[t]}^{r\top}\operatorname{Diag}(\boldsymbol{\gamma}_{[t]}^r) + \beta_{[t]}^r\boldsymbol{k}_{[t]}^r\boldsymbol{k}_{[t]}^{r\top}\sum_{i=1}^{r-1}\operatorname{Diag}\left(\boldsymbol{\gamma}_{[t]}^{i\to r}\right)\boldsymbol{k}_{[t]}^i\boldsymbol{w}_{[t]}^{i\top}$$

$$= \operatorname{Diag}(\boldsymbol{\gamma}_{[t]}^r) - \sum_{i=1}^{r-1}\operatorname{Diag}\left(\boldsymbol{\gamma}_{[t]}^{i\to r}\right)\boldsymbol{k}_{[t]}^i\boldsymbol{w}_{[t]}^{i\top} - \boldsymbol{k}_{[t]}^r\left(\beta_{[t]}^r\operatorname{Diag}(\boldsymbol{\gamma}_{[t]}^r)\boldsymbol{k}_{[t]}^r\right)^\top + \boldsymbol{k}_{[t]}^r\left(\beta_{[t]}^r\sum_{i=1}^{r-1}\boldsymbol{w}_{[t]}^i\left(\boldsymbol{k}_{[t]}^{i\top}\operatorname{Diag}\left(\boldsymbol{\gamma}_{[t]}^{i\to r}\right)\boldsymbol{k}_{[t]}^r\right)\right)^\top$$

$$= \operatorname{Diag}(\boldsymbol{\gamma}_{[t]}^r) - \sum_{i=1}^{r-1}\operatorname{Diag}\left(\boldsymbol{\gamma}_{[t]}^{i\to r}\right)\boldsymbol{k}_{[t]}^i\boldsymbol{w}_{[t]}^{i\top} - \boldsymbol{k}_{[t]}^r\underbrace{\left(\beta_{[t]}^r\left(\operatorname{Diag}(\boldsymbol{\gamma}_{[t]}^r)\boldsymbol{k}_{[t]}^r - \sum_{i=1}^{r-1}\boldsymbol{w}_{[t]}^i\left(\boldsymbol{k}_{[t]}^{i\top}\operatorname{Diag}\left(\boldsymbol{\gamma}_{[t]}^{i\to r}\right)\boldsymbol{k}_{[t]}^r\right)\right)\right)^\top}_{\boldsymbol{w}_{[t]}^r}$$

$$= \operatorname{Diag}(\boldsymbol{\gamma}_{[t]}^r) - \sum_{i=1}^{r-1}\operatorname{Diag}\left(\boldsymbol{\gamma}_{[t]}^{i\to r}\right)\boldsymbol{k}_{[t]}^i\boldsymbol{w}_{[t]}^{i\top} - \boldsymbol{k}_{[t]}^r\boldsymbol{w}_{[t]}^{r\top}$$

$$= \operatorname{Diag}(\boldsymbol{\gamma}_{[t]}^r) - \sum_{i=1}^{r}\operatorname{Diag}\left(\boldsymbol{\gamma}_{[t]}^{i\to r}\right)\boldsymbol{k}_{[t]}^i\boldsymbol{w}_{[t]}^{i\top}$$

The inductive step holds.   ∎

Similar to $\mathbf{P}_{[t]}^r$, $\mathbf{H}_{[t]}^r$ can also be expressed in a parallelizable form.

Proposition 2. *The matrix* $\mathbf{H}^r_{[t]}$ *can be expressed as:*

$$\mathbf{H}^r_{[t]} = \sum_{i=1}^{t} \mathrm{Diag}\left(\boldsymbol{\gamma}^{i \to r}_{[t]}\right) \boldsymbol{k}^i_{[t]} \boldsymbol{u}^{i\top}_{[t]} \tag{18}$$

where the auxiliary vector $\boldsymbol{u}^r_{[t]} \in \mathbb{R}^{d_v}$ *is computed via the following recurrence relation:*

$$\boldsymbol{u}^r_{[t]} = \beta^r_{[t]} \left(\boldsymbol{v}^r_{[t]} - \sum_{i=1}^{r-1} \boldsymbol{u}^i_{[t]} \left(\boldsymbol{k}^{i\top}_{[t]} \mathrm{Diag}\left(\boldsymbol{\gamma}^{i \to r}_{[t]}\right) \boldsymbol{k}^r_{[t]} \right) \right) \tag{19}$$

Proof. We again use mathematical induction.

Inductive Step: Assume the proposition holds for $r - 1$.

$$\mathbf{H}^r_{[t]} = \left(\mathbf{I} - \beta^r_{[t]} \boldsymbol{k}^r_{[t]} \boldsymbol{k}^{r\top}_{[t]}\right) \mathrm{Diag}(\boldsymbol{\alpha}^r_{[t]}) \mathbf{H}^{r-1}_{[t]} + \beta^r_{[t]} \boldsymbol{k}^r_{[t]} \boldsymbol{v}^{r\top}_{[t]}$$

$$= \left(\mathbf{I} - \beta^r_{[t]} \boldsymbol{k}^r_{[t]} \boldsymbol{k}^{r\top}_{[t]}\right) \mathrm{Diag}(\boldsymbol{\alpha}^r_{[t]}) \left(\sum_{i=1}^{r-1} \mathrm{Diag}\left(\boldsymbol{\gamma}^{i \to r-1}_{[t]}\right) \boldsymbol{k}^i_{[t]} \boldsymbol{u}^{i\top}_{[t]}\right) + \beta^r_{[t]} \boldsymbol{k}^r_{[t]} \boldsymbol{v}^{r\top}_{[t]}$$

$$= \left(\mathbf{I} - \beta^r_{[t]} \boldsymbol{k}^r_{[t]} \boldsymbol{k}^{r\top}_{[t]}\right) \left(\sum_{i=1}^{r-1} \mathrm{Diag}\left(\boldsymbol{\gamma}^{i \to r}_{[t]}\right) \boldsymbol{k}^i_{[t]} \boldsymbol{u}^{i\top}_{[t]}\right) + \beta^r_{[t]} \boldsymbol{k}^r_{[t]} \boldsymbol{v}^{r\top}_{[t]}$$

$$= \sum_{i=1}^{r-1} \mathrm{Diag}\left(\boldsymbol{\gamma}^{i \to r}_{[t]}\right) \boldsymbol{k}^i_{[t]} \boldsymbol{u}^{i\top}_{[t]} - \beta^r_{[t]} \boldsymbol{k}^r_{[t]} \boldsymbol{k}^{r\top}_{[t]} \sum_{i=1}^{r-1} \mathrm{Diag}\left(\boldsymbol{\gamma}^{i \to r}_{[t]}\right) \boldsymbol{k}^i_{[t]} \boldsymbol{u}^{i\top}_{[t]} + \beta^r_{[t]} \boldsymbol{k}^r_{[t]} \boldsymbol{v}^{r\top}_{[t]}$$

$$= \sum_{i=1}^{r-1} \mathrm{Diag}\left(\boldsymbol{\gamma}^{i \to r}_{[t]}\right) \boldsymbol{k}^i_{[t]} \boldsymbol{u}^{i\top}_{[t]} - \boldsymbol{k}^r_{[t]} \left(\beta^r_{[t]} \sum_{i=1}^{r-1} \left(\boldsymbol{k}^{r\top}_{[t]} \mathrm{Diag}\left(\boldsymbol{\gamma}^{i \to r}_{[t]}\right) \boldsymbol{k}^i_{[t]}\right) \boldsymbol{u}^i_{[t]}\right)^\top + \boldsymbol{k}^r_{[t]} \beta^r_{[t]} \boldsymbol{v}^{r\top}_{[t]}$$

$$= \sum_{i=1}^{r-1} \mathrm{Diag}\left(\boldsymbol{\gamma}^{i \to r}_{[t]}\right) \boldsymbol{k}^i_{[t]} \boldsymbol{u}^{i\top}_{[t]} + \boldsymbol{k}^r_{[t]} \left(\underbrace{\beta^r_{[t]} \left(\boldsymbol{v}^r_{[t]} - \sum_{i=1}^{r-1} \boldsymbol{u}^i_{[t]} \left(\boldsymbol{k}^{i\top}_{[t]} \mathrm{Diag}\left(\boldsymbol{\gamma}^{i \to r}_{[t]}\right) \boldsymbol{k}^r_{[t]}\right)\right)}_{\boldsymbol{u}^r_{[t]}}\right)^\top$$

$$= \sum_{i=1}^{r-1} \mathrm{Diag}\left(\boldsymbol{\gamma}^{i \to r}_{[t]}\right) \boldsymbol{k}^i_{[t]} \boldsymbol{u}^{i\top}_{[t]} + \boldsymbol{k}^r_{[t]} \boldsymbol{u}^{r\top}_{[t]}$$

$$= \sum_{i=1}^{r} \mathrm{Diag}\left(\boldsymbol{\gamma}^{i \to r}_{[t]}\right) \boldsymbol{k}^i_{[t]} \boldsymbol{u}^{i\top}_{[t]}$$

The inductive step holds.   ∎

C   Pseudo Code for chunkwise KDA

```python
def chunk_kda(
    q: torch.Tensor,
    k: torch.Tensor,
    v: torch.Tensor,
    g: torch.Tensor,
    beta: torch.Tensor,
    initial_state: Optional[torch.Tensor] = None,
    chunk_size: int = 64
):
    dtype = v.dtype
    B, T, H, K, V, C = *q.shape, v.shape[-1], chunk_size
    N = T // C

    q, k, v, g, beta = map(
        lambda x: rearrange(x, 'b (n c) h ... -> b h n c ...', c=C).to(torch.float),
        [q, k, v, g, beta]
    )
    q = q * K**-0.5
    g = g.cumsum(-2)
    mask = torch.triu(torch.ones(C, C, dtype=torch.bool, device=q.device), diagonal=0)

    A = torch.zeros(B, H, N, C, C, dtype=torch.float, device=q.device)
    for i in range(C):
        k_i = k[..., i, :]
        g_i = g[..., i:i+1, :]
        A[..., i] = torch.einsum('... c d, ... d -> ... c', k * (g - g_i).exp(), k_i)
    A = A * beta[..., None]
    # matrix inverse by forward substitution
    A = -A.masked_fill(mask, 0)
    for i in range(1, C):
        A[..., i, :i] = A[..., i, :i].clone() + (A[..., i, :, None].clone() * A[..., :,
        ↪    :i].clone()).sum(-2)
    A = (A + torch.eye(C, dtype=torch.float, device=q.device)) * beta[..., None, :]

    w = A @ (g.exp() * k)
    u = A @ v

    S = k.new_zeros(B, H, K, V)
    if initial_state is not None:
        S += initial_state
    o = torch.zeros_like(v)
    # strictly lower triangular
    mask = torch.triu(torch.ones(C, C, dtype=torch.bool, device=q.device), diagonal=1)
    for i in range(0, N):
        # [B, H, C, ...]
        q_i, k_i, u_i, g_i, w_i = q[:, :, i], k[:, :, i], u[:, :, i], g[:, :, i], w[:, :, i]
        A = torch.zeros(B, H, C, C, dtype=torch.float, device=q.device)
        # secondary chunking for numerical stability
        for j in range(C):
            k_j = k[:, :, i, j]
            g_j = g[:, :, i, j:j+1, :]
            A[..., j] = torch.einsum('... c d, ... d -> ... c', q_i * (g_i - g_j).exp(), k_j)
        A = A.masked_fill(mask, 0)
        v_i = u_i - w_i @ S
        o[:, :, i] = (q_i * g_i.exp()) @ S + A @ v_i
        S = S * rearrange(g_i[:, :, -1].exp(), 'b h k -> b h k 1')
        S += rearrange((g_i[:, :, -1:] - g_i).exp() * k_i, 'b h c k -> b h k c') @ v_i
    return rearrange(o, 'b h n c d -> b (n c) h d').to(dtype)
```

Listing 1: Pseudo PyTorch-style code snippet for KDA chunked form.

D Kimi Linear@5.7T results

Following Moonlight, we also trained Kimi Linear with an extended 5.7T token dataset to demonstrate its effectiveness. With $3\times$ sparsity and a new attention architecture design, Kimi Linear consistently outperforms Moonlight across nearly all benchmarks, underscoring the efficacy of the new architecture. The results are shown in Table 8 for base model and Table 9 for instruction tuned model. Moonlight-Instruct was not evaluated ("-") on tasks exceeding its 8K context limit.

Kimi Linear@5.7T obtains a score of 94.8 on RULER at 1M context length. This long context performance reinforces that Kimi Linear is a promising alternative to full-attention architectures, delivering comparable or superior results while potentially offering more efficient resource utilization.

Table 8: Performance of Kimi-Linear-Base and Moonlight-Base across diverse tasks.

	Benchmark	#Shots	Kimi-Linear-Base	Moonlight-Base
	Architecture	-	MoE	MoE
	# Activated Params	-	3B	3B
	# Total Params	-	48B	16B
	Trained Tokens	-	5.7T	5.7T
General	TriviaQA	5-shots	75.2	66.2
	SimpleQA	5-shots	10.1	5.6
	MMLU-Pro	5-shots	54.8	42.4
	MMLU-redux	5-shots	79.7	73.8
	WinoGrande	5-shots	81.5	74.6
	GPQA-Diamond (avg@8)	5-shots	40.4	35.2
Math	MATH	4-shots	58.5	45.3
	GSM8k	8-shots	86.3	77.2
	GSM8k-platinum	8-shots	89.6	79.4
	CMATH	6-shots	85.5	79.6
Code	CRUXEval-I-cot	0-shots	61.0	45.9
	CRUXEval-O-cot	0-shots	67.0	46.6
	LiveCodeBench (v6)	1-shots	20.0	14.3
	EvalPlus	-	64.9	50.3
Chinese	C-Eval	5-shots	83.3	77.6
	CSimpleQA	5-shots	53.5	34.7

Table 9: Performance of Kimi-Linear-Instruct and Moonlight-Instruct across diverse tasks.

	Benchmark	Kimi-Linear-Instruct	Moonlight-Instruct
	Architecture	MoE	MoE
	# Activated Params	3B	3B
	# Total Params	48B	16B
	Trained Tokens	5.7T	5.7T
General	RULER@128k	95.4	-
	RULER@1M	94.8	-
	GPQA-Diamond (Avg@8)	71.7	24.7
	MMLU-Redux (EM)	86.9	66.9
	MMLU-Pro (EM)	72.7	43.8
	FaithJudge (1-Hallu.)	64.2	56.0
Math	AIME 2025 (Avg@64)	58.6	-
	MATH500 (Acc.)	94.6	58.0
	HMMT 2025 (Avg@32)	44.5	-
Code	LiveCodeBench v6 (Pass@1)	45.7	11.9
	OJBench (Pass@1)	14.2	-
	Humaneval[+]	70.9	46.3
	MBPP[+]	72.4	56.3