Kimi Linear：一种富有表现力、高效的注意力架构

Kimi Linear 技术报告

Kimi 团队

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

Abstract

我们介绍了 Kimi Linear，一种混合线性注意力架构，它首次在公平比较下于各种场景中超越了全注意力机制——包括短上下文、长上下文和强化学习（RL）扩展机制。其核心是 Kimi Delta Attention（KDA），这是一种富有表现力的线性注意力模块，它通过更细粒度的门控机制扩展了 Gated DeltaNet [111]，从而更有效地利用有限的有限状态 RNN 内存。我们定制的分块算法通过对角加低秩（DPLR）转移矩阵的专用变体实现了高硬件效率，与通用 DPLR 公式相比大幅减少了计算量，同时与经典 delta 规则保持更高的一致性。

我们预训练了一个具有 30 亿激活参数和 480 亿总参数的 Kimi Linear 模型，基于 KDA 和多头潜在注意力（MLA）的逐层混合架构。我们的实验表明，在相同的训练配方下，Kimi Linear 在所有评估任务上均以显著优势超越了全 MLA，同时将 KV 缓存使用量减少多达 75%，并在 1M 上下文长度下实现高达 6× 的解码吞吐量。这些结果表明，Kimi Linear 可以作为全注意力架构的即插即用替代品，具有更优越的性能和效率，包括处理更长输入和输出长度的任务。

为了支持进一步的研究，我们开源了 KDA 内核和 vLLM 实现[1]，并发布了预训练和指令微调模型检查点。[2]

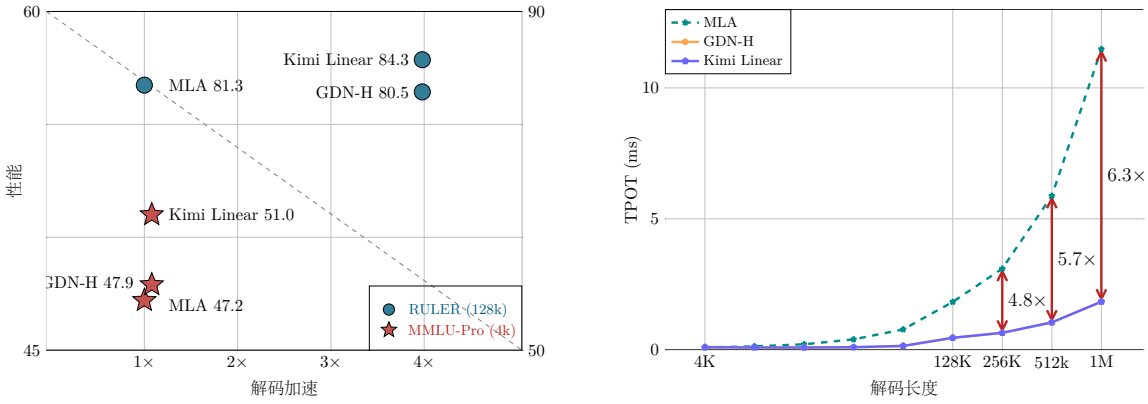
1 引言

随着大型语言模型（LLM）逐渐演变为能力越来越强的智能体 [50]，推理的计算需求——特别是在长程和强化学习（RL）设置中——正在成为核心瓶颈。这种向 *RL 测试时扩展* [95, 33, 80, 74, 53] 的转变，其中模型必须在推理时处理扩展轨迹、工具使用交互和复杂决策空间，暴露了标准注意力机制的根本低效。特别是，softmax 注意力的二次时间复杂度和线性增长的键值（KV）缓存引入了巨大的计算和内存开销，阻碍了吞吐量、上下文长度扩展和实时交互性。

线性注意力 [48] 提供了一种降低计算复杂度的原则性方法，但由于表达能力有限，历史上在语言建模方面表现不如 softmax 注意力——即使对于短序列也是如此。最近的进展显著缩小了这一差距，主要通过两项创新：门控或衰减机制 [92, 16, 114] 和 delta 规则 [84, 112, 111, 71]。这些发展共同推动线性

[1] https://github.com/fla-org/flash-linear-attention/tree/main/fla/ops/kda

[2] https://huggingface.co/moonshotai/Kimi-Linear-48B-A3B-Instruct



(a) (b)

图 1: (a) 性能与加速对比。在 1.4T 训练令牌的严格公平比较下，在 MMLU-Pro（4k 上下文长度，红星）上，Kimi Linear 在相似速度下领先性能（51.0）。在 RULER（128k 上下文长度，蓝圈）上，它是帕累托最优的，实现顶级性能（84.3）和 3.98× 加速。(b) 每输出令牌时间（TPOT）与解码长度对比。Kimi Linear（蓝线）保持低 TPOT，与 GDN-H 匹配，在长序列上超越 MLA。这使得更大的批次成为可能，在 1M 令牌处产生比 MLA 快 6.3× 的 TPOT（1.84ms vs. 11.48ms）。

注意力在中等长度序列上接近 softmax 级别的质量。然而，纯线性结构仍然受到有限状态容量的根本限制，使得长序列建模和上下文内检索在理论上具有挑战性 [104, 4, 45]。

结合 softmax 和线性注意力的混合架构——使用少数全局注意力层与主要更快的线性层相结合——因此成为质量与效率之间实用折衷 [57, 100, 66, 12, 32, 81]。然而，以前的混合模型通常在有限规模下运行或缺乏跨多样化基准的全面评估。核心挑战仍然存在：开发一种在质量上匹配或超越全注意力，同时在速度和内存方面实现实质性效率提升的注意力架构——这是实现下一代智能体、解码密集型 LLM 的关键一步。

在这项工作中，我们提出了 **Kimi Linear**，一种混合线性注意力架构，旨在满足智能体智能和测试时扩展的效率需求，同时不牺牲质量。其核心是 **Kimi Delta Attention（KDA）**，一种硬件高效的线性注意力模块，它通过更细粒度的门控机制扩展了 Gated DeltaNet [111]。虽然 GDN 与 Mamba2 [16] 类似，采用粗略的头级遗忘门，但 KDA 引入了通道级变体，其中每个特征维度保持独立的遗忘率，类似于门控线性注意力（GLA） [114]。这种细粒度设计能够更精确地调节有限状态 RNN 内存，释放了混合架构中 RNN 风格模型的潜力。

至关重要的是，KDA 使用对角加低秩（DPLR）矩阵 [30, 71] 的专用变体对其转移动态进行参数化，实现了一个定制的分块并行算法，相对于通用 DPLR 公式大幅减少了计算量，同时与经典 delta 规则保持一致。

Kimi Linear 以均匀的 3:1 比例将 KDA 与周期性全注意力层交错排列。这种混合结构在长序列生成期间将内存和 KV 缓存使用量减少多达 75%，同时通过全注意力层保持全局信息流。通过匹配规模的预训练和评估，我们表明 Kimi Linear 在短上下文、长上下文和 RL 风格后训练任务上一致匹配或超越强大的全注意力基线——同时在 1M 上下文长度下实现高达 6× 的更高解码吞吐量。

为了促进进一步的研究，我们发布了开源 KDA 内核与 vLLM 集成，以及预训练和指令微调检查点。这些组件与现有的全注意力管道即插即用兼容，无需修改缓存或调度接口，从而促进混合架构的研究。

贡献

- **Kimi Delta Attention（KDA）**：一种线性注意力机制，通过改进的循环内存管理和硬件效率来细化门控 delta 规则。
- **Kimi Linear 架构**：一种混合设计，采用 3:1 的 KDA 与全局注意力比例，在超越全注意力质量的同时减少内存占用。
- **规模上的公平实证验证**：通过 1.4T 令牌训练运行，Kimi Linear 在短/长上下文和 RL 风格评估中超越全注意力和其他基线，并完全发布内核、vLLM 集成和检查点。

2 预备知识

在本节中，我们介绍与提出的 Kimi Delta Attention 相关的技术背景。

2.1 符号说明

在本文中，我们定义 $\square_t \in \mathbb{R}^{d_k}$ 或 $\mathbb{R}^{d_v}$，s. t.，$\square \in \{\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v}, \boldsymbol{o}, \boldsymbol{u}, \boldsymbol{w}\}$ 表示第 t 个相应的列向量，$\mathbf{S}_t \in \mathbb{R}^{d_k \times d_v}$ 表示矩阵形式的记忆状态。$\mathbf{M}$ 和 $\mathbf{M}^-$ 分别表示包含和不包含对角元素的下三角掩码；为方便起见，我们也将它们写作 Tril 和 StrictTril。

分块形式 假设序列被分成 L/C 个块，每个块长度为 C。我们定义 $\square_{[t]} \in \mathbb{R}^{C \times d}$（其中 $\square \in \{\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{O}, \mathbf{U}, \mathbf{W}\}$）为在第 t 个块内向量堆叠而成的矩阵，$\square_{[t]}^r = \square_{tC+r}$ 是块中的第 r 个元素。注意 $t \in [0, L/C), r \in [1, C]$。状态矩阵也重新索引为 $\boldsymbol{S}_{[t]}^i = \boldsymbol{S}_{tC+i}$。此外，$\boldsymbol{S}_{[t]} := \boldsymbol{S}_{[t]}^0 = \boldsymbol{S}_{[t-1]}^C$，即块的初始状态是前一个块的最后一个状态。

衰减形式 我们定义累积衰减 $\gamma_{[t]}^{i \to j} := \prod_{k=i}^{j} \alpha_{[t]}^k$，并将 $\gamma_{[t]}^{1 \to r}$ 简写为 $\gamma_{[t]}^r$。此外，$\mathcal{A}_{[t]} := \mathcal{A}_{[t]}^{i/j} \in \mathbb{R}^{C \times C}$ 是元素为 $\gamma_{[t]}^i / \gamma_{[t]}^j$ 的矩阵。$\mathrm{Diag}\left(\boldsymbol{\alpha}_t\right)$ 表示细粒度衰减，$\mathrm{Diag}\left(\boldsymbol{\gamma}_{[t]}^{i \to j}\right) := \prod_{k=i}^{j} \mathrm{Diag}\left(\boldsymbol{\alpha}_{[t]}^k\right)$，$\boldsymbol{\Gamma}_{[t]}^{i \to j} \in \mathbb{R}^{C \times d_k}$ 是从 $\boldsymbol{\gamma}_{[t]}^i$ 到 $\boldsymbol{\gamma}_{[t]}^j$ 堆叠的矩阵。

2.2 线性注意力与门控 Delta 规则

作为在线学习的线性注意力 线性注意力 [48] 维护一个矩阵值循环状态，累积键值关联：

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

从快速权重视角 [84, 85]，$\mathbf{S}_t$ 作为关联记忆存储从键到值的瞬态映射。这种更新可以看作是在无界相关目标上执行梯度下降

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

它不断强化最近的键值对而没有任何遗忘。然而，这种目标没有提供关于应该擦除哪些记忆的标准，累积状态无界增长，导致长上下文中的干扰。

DeltaNet：在重构损失上的在线梯度下降 DeltaNet [84] 将这种递归重新解释为在重构目标上的在线梯度下降：

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

使用学习率 β_t 进行梯度步骤得到

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

这个规则——经典的 *delta* 规则——将 $\mathbf{S}$ 视为可学习的关联记忆，不断向映射 $\boldsymbol{k}_t \mapsto \boldsymbol{v}_t$ 自我纠正。秩-1 更新结构，等价于广义 Householder 变换，支持硬件高效的分块并行化 [11, 112]。

作为权重衰减的门控 DeltaNet 尽管 DeltaNet 稳定了学习，但它仍然无限期地保留过时的关联。门控 DeltaNet（GDN） [111] 引入标量遗忘门 $\alpha_t \in [0, 1]$，得到

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

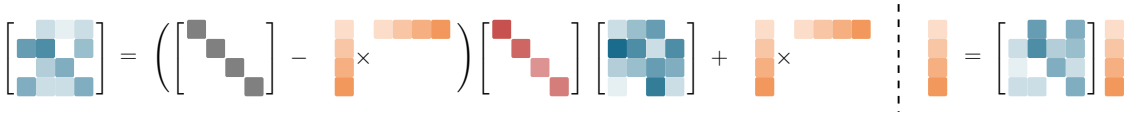
这里，α_t 作为快速权重上的权重衰减形式 [8]，实现类似于数据依赖 L_2 正则化的遗忘机制。这种简单而有效的修改为控制记忆寿命和减轻干扰提供了原则性方法，提高了稳定性和长上下文泛化能力，同时保留了 DeltaNet 的可并行化结构。

从这个角度来看，我们观察到 GDN 可以被解释为一种乘法位置编码形式，其中转移矩阵是数据依赖且可学习的，放松了 RoPE [115] 的正交性约束。[3]

3 Kimi Delta Attention：用细粒度门控改进 Delta 规则

我们提出了 Kimi Delta Attention（KDA），一种新的门控线性注意力变体，它通过引入细粒度对角门 $\mathrm{Diag}(\boldsymbol{\alpha}_t)$ 来细化 GDN 的标量衰减，实现对记忆衰减和位置感知的细粒度控制（如 §6.1所讨论）。我们首先介绍 KDA 的分块并行化，展示如何将一系列秩-1 矩阵变换压缩为密集表示，同时在对角门控下保持稳定性。然后我们强调 KDA 相对于标准 DPLR（对角加低秩）公式 [30, 71] 的效率提升。

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



3.1 硬件高效的分块算法

通过将公式1的递归部分展开为分块形式，我们有：

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

[3] 当状态变换矩阵保持其正交性时，绝对位置编码也可以独立应用于 $\boldsymbol{q}$ 和 $\boldsymbol{k}$，在注意力计算期间转换为相对位置编码 [87]。

WY 表示 WY 表示通常用于将一系列秩-1 更新打包为单个紧凑表示 [11]。我们遵循 Comba [40] 中 **P** 的公式，以减少后续计算中额外矩阵求逆的需要。

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

其中辅助向量 $\boldsymbol{w}_t \in \mathbb{R}^{d_k}$ 和 $\boldsymbol{u}_t \in \mathbb{R}^{d_v}$ 通过以下递推关系计算：

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

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

UT 变换 我们应用 UT 变换 [46] 来减少非矩阵乘法 FLOPs，这对于在训练期间实现更好的硬件利用率至关重要。

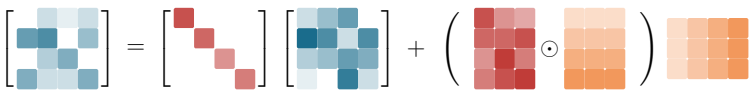
$$\mathbf{M}_{[t]} = \left(\mathbf{I} + \mathrm{StrictTril}\left(\mathrm{Diag}\left(\beta_{[t]}\right)\left(\boldsymbol{\Gamma}^{1\to C}_{[t]} \odot \mathbf{K}_{[t]}\right)\left(\frac{\mathbf{K}_{[t]}}{\boldsymbol{\Gamma}^{1\to C}_{[t]}}\right)^{\top}\right)\right)^{-1}\mathrm{Diag}\left(\beta_{[t]}\right)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

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

下三角矩阵的逆可以通过高斯消元中的前向替换迭代按行高效计算 [28]。

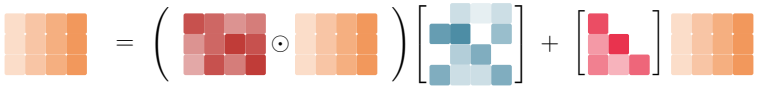
等价地，以矩阵形式，我们可以按块更新状态：

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



在输出阶段，我们采用块间循环和块内并行策略，以最大化矩阵乘法吞吐量，从而充分利用 Tensor Cores 的计算潜力。

$$\mathbf{O}_{[t]} = \underbrace{\left(\boldsymbol{\Gamma}^{1\to C}_{[t]} \odot \mathbf{Q}_{[t]}\right)\mathbf{S}_{[t]}}_{\text{块间}} + \underbrace{\mathrm{Tril}\left(\left(\boldsymbol{\Gamma}^{1\to C}_{[t]} \odot \mathbf{Q}_{[t]}\right)\left(\frac{\mathbf{K}_{[t]}}{\boldsymbol{\Gamma}^{1\to C}_{[t]}}\right)^{\top}\right)}_{\text{块内}}\underbrace{\left(\mathbf{U}_{[t]} - \mathbf{W}_{[t]}\mathbf{S}_{[t]}\right)}_{\text{“伪”值项}} \in \mathbb{R}^{C\times d_v} \tag{9}$$



3.2 效率分析

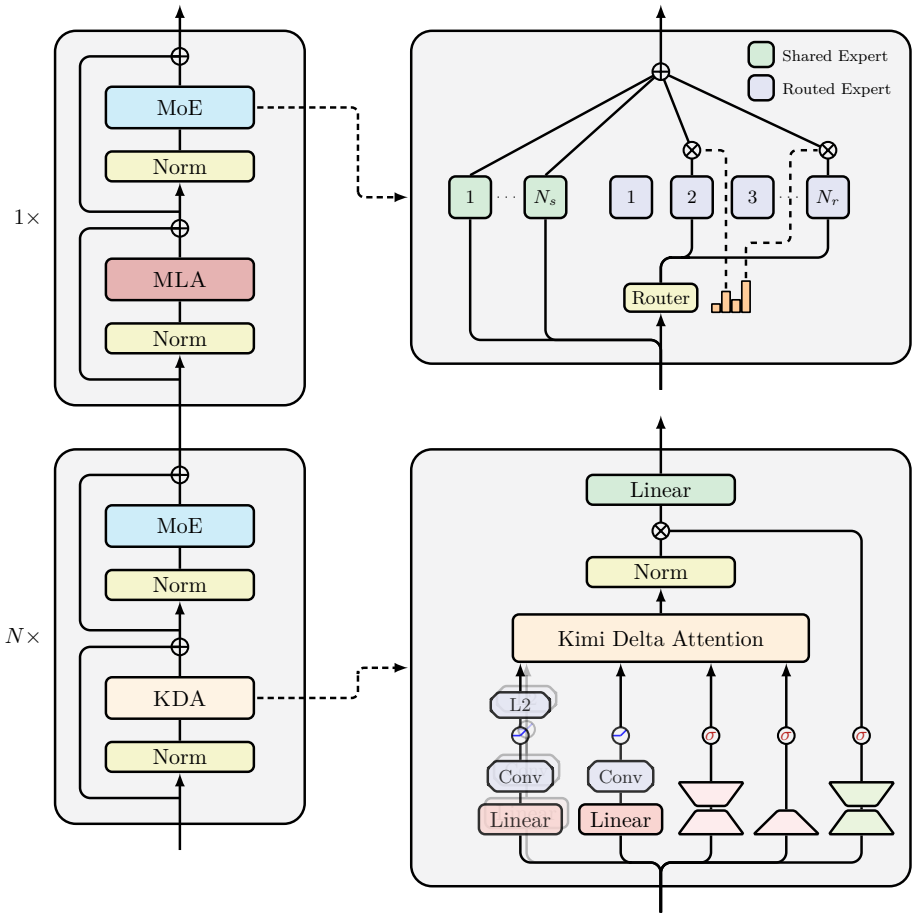


图 3: Kimi Linear 模型架构示意图，由包含令牌混合层和 MoE 通道混合层的块堆叠而成。具体来说，我们将 N 个 KDA 层与 1 个 MLA 层交错用于令牌混合，其中 N 在我们的实现中设置为 3。

在表示能力方面，KDA 与广义 DPLR 公式对齐，即 $\mathbf{S}_t = (\mathbf{D} - \boldsymbol{a}_t\boldsymbol{b}_t^\top)\mathbf{S}_{t-1} + \boldsymbol{k}_t\boldsymbol{v}_t^\top$，两者都表现出细粒度衰减行为。然而，这种细粒度衰减在除法运算期间引入数值精度问题（例如公式9中的块内计算）。为解决此问题，GLA 等先前工作 [114] 在对数域执行计算并引入全精度的二级分块。然而，这种方法阻止了半精度矩阵乘法的充分利用，显著降低了算子速度。通过将变量 $\boldsymbol{a}$ 和 $\boldsymbol{b}$ 都绑定到 $\boldsymbol{k}$，KDA 有效缓解了这一瓶颈——将二级块矩阵计算的数量从四个减少到两个，并进一步消除了三个额外的矩阵乘法。因此，KDA 的算子效率相比 DPLR 公式提高了约 100%。详细分析见 §6.2。

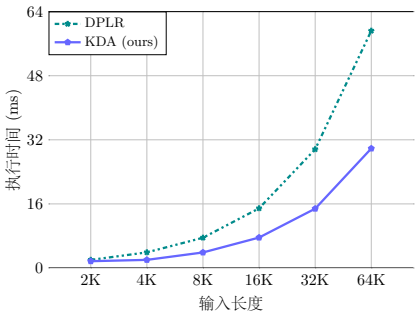


图 2: 不同输入长度的内核执行时间，统一批次大小为 1，16 个头。

4 Kimi Linear 模型架构

我们模型架构的主干遵循 Moonlight [62]。除了细粒度门控之外，我们还利用了几个组件来进一步提高 Kimi Linear 的表达能力。Kimi Linear 的整体架构如图3所示。

神经参数化 令 $\boldsymbol{x}_t \in \mathbb{R}^d$ 为第 t 个令牌输入表示，每个头 h 的 KDA 输入计算如下

$$\begin{aligned}
\boldsymbol{q}_t^h, \boldsymbol{k}_t^h &= \mathrm{L2Norm}(\mathrm{Swish}(\mathrm{ShortConv}(\mathbf{W}_{q/k}^h \boldsymbol{x}_t))) \in \mathbb{R}^{d_k} \\
\boldsymbol{v}_t^h &= \mathrm{Swish}(\mathrm{ShortConv}(\mathbf{W}_v^h \boldsymbol{x}_t)) \in \mathbb{R}^{d_v} \\
\boldsymbol{\alpha}_t^h &= f(\mathbf{W}_\alpha^\uparrow \mathbf{W}_\alpha^\downarrow \boldsymbol{x}_t) \in [0,1]^{d_k} \\
\beta_t^h &= \mathrm{Sigmoid}(\mathbf{W}_\beta^h \boldsymbol{x}_t) \in [0,1]
\end{aligned}$$

其中 d_k, d_v 表示键和值的头维度，所有实验中都设置为 128。对于 $\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v}$，我们应用 ShortConv 后跟 Swish 激活，遵循 [111]。$\boldsymbol{q}$ 和 $\boldsymbol{k}$ 表示进一步使用 L2Norm 归一化以确保特征值稳定性，如 [112] 所建议。每通道衰减 $\boldsymbol{\alpha}_t^h$ 通过低秩投影（$\mathbf{W}_\alpha^\downarrow$ 和 $\mathbf{W}_\alpha^\uparrow$，秩等于头维度）和衰减函数 $f(\cdot)$ 参数化，类似于 GDN 和 Mamba 中使用的函数 [111, 16]。在通过 $\mathbf{W}_o \in \mathbb{R}^{d \times d}$ 进行输出投影之前，我们使用头级 RMSNorm [122] 和数据依赖门控机制 [79]，参数化为：

$$\boldsymbol{o}_t = \mathbf{W}_o \left(\mathrm{Sigmoid}\left(\mathbf{W}_g^\uparrow \mathbf{W}_g^\downarrow \boldsymbol{x}_t \right) \odot \mathrm{RMSNorm}\left(\mathrm{KDA}\left(\boldsymbol{q}_t, \boldsymbol{k}_t, \boldsymbol{v}_t, \boldsymbol{\alpha}_t, \beta_t \right) \right) \right) \tag{10}$$

这里，输出门采用与遗忘门类似的低秩参数化，以确保公平的参数比较，同时保持与全秩门控相当的性能并缓解注意力汇聚 [79]。非线性激活函数的选择在 §5.2中进一步讨论。

混合模型架构 长上下文检索仍然是纯线性注意力的主要瓶颈，因此我们将 KDA 与少量全全局注意力（Full MLA）层 [19] 混合。对于 Kimi Linear，我们选择逐层方法（交替整个层）而不是头级方法（在层内混合头），因为前者具有更优的基础设施简单性和训练稳定性。经验上，均匀的 3:1 比例，即 3 个 KDA 层对应 1 个全 MLA 层，提供了最佳的质量-吞吐量权衡。我们在 § 7.2中讨论其他混合策略。

MLA 层的无位置编码（NoPE） 在 Kimi Linear 中，我们对所有全注意力（MLA）层应用 NoPE。这种设计将整个位置信息编码和近因偏置（见 § 6.1）的责任委托给 KDA 层。因此，KDA 被确立为主要的位置感知算子，履行类似于或 arguably 强于短卷积 [3] 或 SWA [76] 等辅助组件的角色。我们的发现与先前结果 [110, 7, 19] 一致，他们同样证明用专门的位置感知机制补充全局 NoPE 注意力可以产生有竞争力的长上下文性能。

我们注意到 NoPE 提供了实际优势，特别是对于 MLA。首先，NoPE 使其能够在推理期间转换为高效的纯多查询注意力（MQA）。其次，它简化了长上下文训练，因为它消除了对 RoPE 参数调整的需要，如频率基调优或 YaRN 等方法 [72]。

5 实验

5.1 合成测试

我们首先评估 KDA 与其他竞争性线性注意力方法在三个合成任务上的性能，作为长上下文性能的基准测试。在所有实验中，我们采用一致的模型配置：2 层，2 个注意力头，每个头维度为 128。对于每个任务，我们使用学习率网格搜索 $\{5\times10^{-5}, 1\times10^{-4}, 5\times10^{-4}, 1\times10^{-3}\}$ 训练模型最多 20,000 步。然后我们展示表现最佳的训练准确率曲线。具体来说，我们比较两种场景：（1）不同任务在训练长度从 256 增加到 2,048 个令牌时的性能，测量峰值准确率；（2）KDA、GDN 和 Mamba2 在固定上下文长度 1,024 个令牌时的收敛速度。

回文 回文要求模型按相反顺序再现给定的随机令牌序列。如表5.1所示，给定像 “O G R S U N E” 这样的输入，模型必须生成其精确反转。这种复制任务对于线性注意力模型来说已知是困难的 [45]，因为它们难以从压缩的固定大小记忆状态中精确检索整个历史。

输入	O	G	R	S	U	N	E	<SEP>	E	N	U	S	R	G	O
输出	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	N	U	S	R	G	O	ϕ

多查询关联回忆（MQAR） MQAR 评估模型检索与出现在上下文各位置的多个查询相关联的值的能力。例如，如表5.1所示，模型被要求回忆查询B对应的0和G对应的5。这项任务已知与语言建模性能高度相关 [5]。

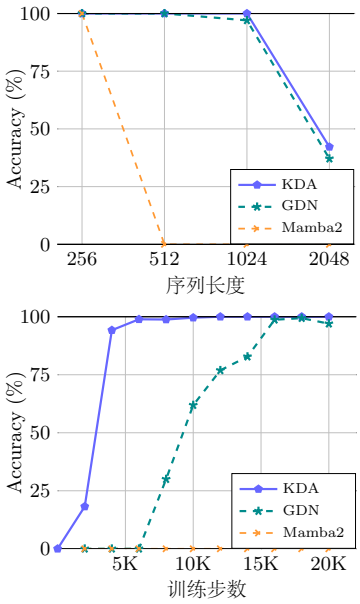
输入	A	1	C	3	B	0	M	8	G	5	E	4	<SEP>	B	G
输出	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	0	5

栈 我们通过模拟标准 LIFO（后进先出）栈操作来评估每个候选者的状态跟踪能力 [27]。我们的设置涉及 64 个独立的栈，每个栈由唯一 ID 标识。模型处理两种操作序列：1）PUSH：像 “<PUSH> 1 G” 这样的操作将元素G添加到栈1；2）POP：像 “<POP> 0 E” 这样的操作要求模型预测最近推入栈0的元素E。目标是准确跟踪所有栈的状态并在每次弹出请求时预测正确的元素。

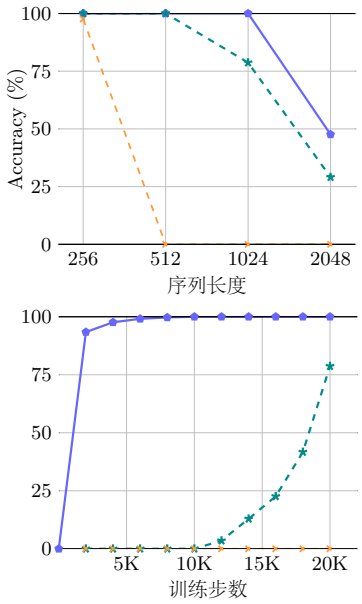
图4显示了最终结果。在所有任务中，当序列长度从 256 增加到 2,048 个令牌时，KDA 始终实现最高准确率。特别是在回文和回忆密集型 MQAR 任务上，KDA 比 GDN 收敛得显著更快。这证实了我们细粒度衰减的好处，它使模型能够有选择地遗忘不相关信息，同时更精确地保留关键记忆。我们还观察到，Mamba2 [16] 是一种典型的线性注意力，仅使用乘法衰减而缺乏 delta 规则，在我们的模型设置下在所有任务上都失败了。

5.2 Kimi Linear 关键组件的消融研究

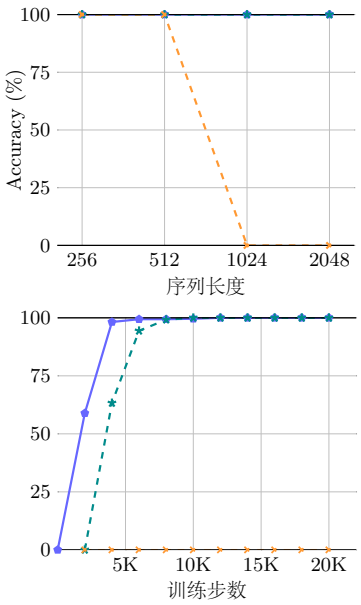
我们通过直接将不同模型与第一规模缩放定律模型（即 16 个头，16 层）进行比较来进行一系列消融研究。所有模型都使用相同的 FLOPs 预算和超参数进行训练以确保公平比较。我们在表1中报告训练和验证困惑度（PPL）。验证 PPL 在与预训练语料库分布显著不同的高质量数据集上计算，强调分布偏移下的泛化，因此训练和验证困惑度存在差异。



(a) 回文



(b) 多查询关联回忆



(c) 栈

图 4: 合成任务的结果：回文、多查询关联回忆和状态跟踪。

输出门 我们将默认的 Sigmoid 输出门与两个变体进行比较：一个无门控，另一个使用 swish 门控。结果表明，移除门控会降低性能。此外， [111] 采用的 swish 门控表现明显差于 Sigmoid。我们的观察与 [79] 一致，他们也得出结论 Sigmoid 门控提供卓越的性能。因此我们在所有实验中采用 Sigmoid，包括 GDN-H。

卷积层 轻量级深度卷积，小核大小（例如 4）可以有效捕获局部令牌依赖 [3]，并被许多最近架构广泛采用 [16, 5, 112]。我们在表1中验证其功效，证明卷积层在混合模型中继续发挥不可忽视的作用。

表 1: 对 KDA 与 MLA 注意力混合比例及其他关键组件的消融研究。我们列出训练和验证困惑度（越低越好）进行比较。我们最终实验中使用的最佳性能模型以灰色突出显示。

		训练 PPL (↓)	验证 PPL (↓)
混合比例	3:1	**9.23**	**5.65**
	0:1	9.45	5.77
	1:1	9.29	5.66
	7:1	9.23	5.70
	15:1	9.34	5.82
无输出门		9.25	5.67
使用 swish 输出门		9.43	5.81
无卷积层		9.29	5.70

表 2: 缩放定律实验的模型配置和超参数。

# 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.

混合比例 我们进行了消融研究以确定 KDA 线性注意力层与 MLA 全注意力层的最优混合比例。在测试的配置中，3:1 比例（每 1 个 MLA 层对应 3 个 KDA 层）产生了最佳结果，实现了最低的训练和验证损失。我们观察到其他比例的明显权衡：更高比例（例如 7:1）产生相当的训练损失，但导致显著更差的验证性能，而更低比例（例如 1:1）保持相似的验证损失，但以增加推理开销为代价。此外，纯全注意力基线（0:1）表现不佳。因此，3:1 配置提供了模型性能和计算效率之间最有效的平衡。

NoPE vs. RoPE 如表5所示，Kimi Linear 在长上下文评估中始终表现出色，而 Kimi Linear（RoPE）在短上下文任务上获得相似分数。我们认为这种差异源于位置偏置在深度上的分布方式。在 Kimi Linear（RoPE）中，全局注意力层携带强烈的显式相对位置信号，而线性注意力（例如 GDN）贡献较弱的隐式位置归纳偏置。这种不匹配导致全局层对短程顺序的过度强调，有利于短上下文，但使模型在训练中期适应扩展上下文时灵活性降低。相比之下，Kimi Linear 在各层之间诱导更平衡的位置偏置，提高鲁棒性和长距离外推能力，从而产生更强的长上下文性能。关于长上下文性能，如表5所示，Kimi Linear 在不同长上下文基准上实现最佳平均分数，验证了我们上一节中声称的好处。

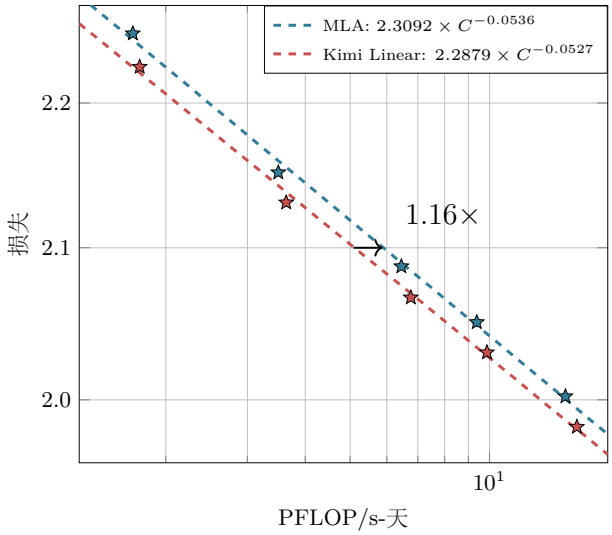


图 5: MLA 和 Kimi Linear 的拟合缩放定律曲线。

5.3 Kimi Linear 的缩放定律

我们在一系列遵循 Moonlight [62] 架构的 MoE 模型上进行了缩放定律实验。在所有实验中，我们激活 64 个专家中的 8 个，并使用 Muon 优化器 [62]。详细信息和超参数列于表2。

对于 MLA，遵循 Chinchilla 缩放定律方法 [37]，我们训练了五种不同规模的语言模型，通过网格搜索仔细调整其超参数以确保每个模型的最佳性能。对于 KDA，我们保持表1中消融的最佳混合比例 3:1。除此之外，我们严格遵守 MLA 训练配置，不做任何修改。如图5所示，Kimi Linear 在计算最优训练下相比 MLA 基线实现了 $\sim 1.16\times$ 的计算效率。我们期望仔细的超参数调整将为 KDA 产生更优越的缩放曲线。

5.4 实验设置

Kimi Linear 和基线设置 我们评估我们的 Kimi Linear 模型与全注意力 MLA 基线和混合门控 DeltaNet（GDN-H）基线，所有模型共享相同的架构、参数数量和训练设置以确保公平比较。模型配置主要与 Moonlight [62] 对齐，关键区别是 MoE 稀疏度增加到 32。每个模型激活 256 个专家中的 8 个，包括一个共享专家，resulting in 480 亿总参数和每次前向传播 30 亿激活参数。第一层实现为没有 MoE 的密集层，确保稳定训练。为评估 Kimi Linear 中 NoPE 的有效性，我们还引入了使用相同模型配置但使用 RoPE 的混合 KDA 基线，称为 Kimi Linear（RoPE）。

评估基准 我们的评估涵盖三大类基准，每类旨在评估模型的不同能力：

- **语言理解与推理**：Hellaswag [121]、ARC-Challenge [14]、Winogrande [83]、MMLU [36]、TriviaQA [47]、MMLU-Redux [26]、MMLU-Pro [103]、GPQA-Diamond [82]、BBH [94] 和 [105]。
- **代码生成**：LiveCodeBench v6 [4] [44]、EvalPlus [60]。

[4] 2024.8 至 2025.5 的问题

- **数学与推理**：AIME 2025、MATH 500、HMMT 2025、PolyMath-en。
- **长上下文**：MRCR [5]、RULER [38]、Frames [52]、HELMET-ICL [118]、RepoQA [61]、Long Code Arena [13] 和 LongBench v2 [6]。
- **中文语言理解与推理**：C-Eval [43] 和 CMMLU [55]。

评估配置 所有模型使用温度 1.0 进行评估。对于高方差基准，我们报告 Avg@k 分数。对于基础模型，我们对 MMLU、MMLU-Redux、GPQA-Diamond 和 C-Eval 采用基于困惑度的评估。否则采用基于生成的评估。为减轻 GPQA-Diamond 固有的高方差，我们报告八次独立运行的平均分数。所有评估使用我们内部源自 LM-Harness-Evaluation [10] 的框架进行，确保所有模型的一致设置。

5.4.1 预训练配方

预训练配方 所有模型使用 4,096 令牌上下文窗口、MuonClip 优化器和 WSD 学习率计划进行预训练，处理从 K2 预训练语料库 [50] 采样的总计 1.4 万亿令牌的共享数据。学习率设置为 1.1×10^{-3}，全局批次大小固定为 3200 万令牌。它们还采用 Kimi K2 [50] 中建立的相同退火计划和长上下文激活阶段。

我们最终发布的 Kimi Linear 检查点使用相同程序进行预训练，但扩展了总计 5.7 万亿令牌以匹配 Moonlight 的预训练令牌。此外，最终检查点支持长达 100 万令牌的上下文长度。我们在附录 D中比较 Kimi Linear@5.7T 和 Moonlight 的性能。

5.4.2 后训练配方

SFT 配方 SFT 数据集通过整合额外的推理任务扩展了 Kimi K2 [50] SFT 数据，创建了一个涵盖多样化领域的大规模指令微调数据集，重点强调数学和编码。我们采用多阶段 SFT 方法，最初在广泛的多样化 SFT 数据上训练模型以实现通用指令遵循，然后对推理密集型数据进行计划性针对性训练以增强模型的推理能力。

RL 配方 对于 RL 训练提示集，我们主要整合三个数据源：数学、代码和 STEM。这种增强的主要目的是提高模型的推理能力。在进行 RL 之前，我们预先选择与中等等难度级别匹配的数据用于起始检查点。

RL 训练的一个已知风险是通用能力的潜在退化。为缓解这一点，我们在 RL 期间纳入 PTX 损失 [70]，遵循 K2 [50] 的做法。这涉及在 RL 过程中对高质量、分布多样化数据集进行并发 SFT。我们的 PTX 数据集涵盖推理和通用任务。上述所有数据均源自 K2 模型 [50] 训练配方的子集。

对于 RL 算法，我们使用与 K1.5 [95] 相同的算法，同时引入了几个高级技巧。我们注意到训练和推理引擎之间的精度不匹配可能导致 RL 学习不稳定。因此我们引入截断重要性采样，一种有效缓解 rollout 和训练之间策略不匹配的方法 [116]。我们还动态调整 KL 惩罚和迷你批次大小（即每次迭代的更新次数）以使 RL 训练稳定并避免熵崩溃 [15]。

[5] https://huggingface.co/datasets/openai/mrcr

5.5 主要结果

5.5.1 Kimi Linear@1.4T 结果

预训练结果 我们在表3中使用 1.4T 预训练语料库将 Kimi Linear 模型与两个基线（MLA 和混合 GDN-H）进行比较。评估侧重于三个领域：通用知识、推理（数学和代码）和中文任务。Kimi Linear 在几乎所有类别上都始终优于两个基线。

- 通用知识：Kimi Linear 在所有关键基准如 BBH、MMLU 和 HellaSwag 上得分最高。
- 推理：它在数学（GSM8K）和大多数代码任务（CRUXEval）上领先。然而，在 EvalPlus 上相比 GDN-H 得分略低。
- 中文任务：Kimi Linear 在 CEval 和 CMMLU 上获得最高分。

总之，Kimi Linear 展示了最强的性能，将其定位为短上下文预训练中全注意力架构的有力替代方案。

表 3: Kimi Linear 与全注意力 MLA 基线和混合 GDN 基线的性能比较，所有模型经过相同的预训练配方。Kimi Linear 在短上下文预训练评估中始终优于 MLA 和 GDN-H。每列最佳结果以**粗体**显示。

	类型基线	MLA	GDN-H	Kimi Linear
	训练令牌	1.4T	1.4T	1.4T
通用	HellaSwag	81.7	82.2	**82.9**
	ARC-challenge	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**
数学与代码	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**
中文	CEval	79.3	79.1	**79.5**
	CMMLU	79.5	80.7	**80.8**

SFT 结果 Kimi Linear 在经过相同的监督微调（SFT）配方后，在通用任务和数学与代码任务上都表现出强劲性能，始终优于 MLA 和 GDN-H。在通用任务中，Kimi Linear 全面领先，在各种 MMLU 基准、BBH 和 GPQA-Diamond 上获得最高分。在数学与代码任务中，它在 AIME 2025、HMMT 2025、PolyMath-en 和 LiveCodeBench 等困难基准上超越两个基线。尽管有一些小的例外如 MATH500 和 EvalPlus，Kimi Linear 在任务上表现出稳健的优越性，确认其相对于其他测试模型（GDN-H 和 MLA）的明显优势。

表 4: Kimi Linear 与全注意力 MLA 基线和混合 GDN 基线的性能比较，所有模型在预训练后使用相同的 SFT 配方。Kimi Linear 在短上下文指令微调基准上始终优于 MLA 和 GDN-H。每列最佳结果以**粗体**显示。

	类型指令微调	MLA	GDN-H	Kimi Linear
	训练令牌	1.4T	1.4T	1.4T
通用	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
数学与代码	AIME 2025 (Avg@64)	20.6	21.1	**21.3**
	MATH500 (准确率)	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

表 5: Kimi Linear 与 MLA、GDN-H 和 Kimi Linear（RoPE）在长上下文基准上的比较。最后一列报告总体平均值(↑)。所有模型训练于 1.4T 令牌。每列最佳结果以**粗体**显示。

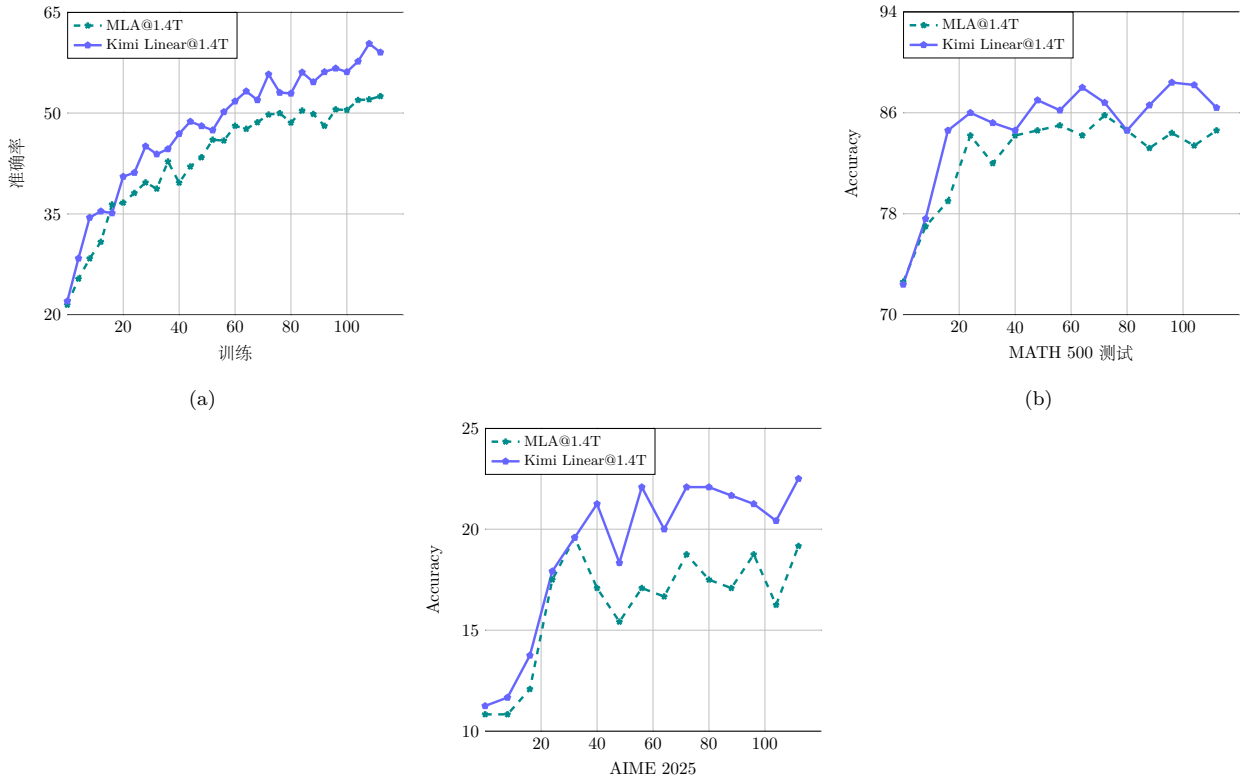
	RULER	MRCR	HELMET-ICL	LongBench V2	Frames	RepoQA	Long Code Arena		平均
							Lib	Commit	
MLA	81.3	22.6	88.0	**36.1**	**60.5**	63.0	32.8	**33.2**	52.2
GDN-H	80.5	23.9	85.5	32.6	58.7	63.0	34.7	30.5	51.2
Kimi Linear (RoPE)	78.8	22.0	88.0	35.4	59.9	66.5	31.3	32.5	51.8
Kimi Linear	**84.3**	**29.6**	**90.0**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

长上下文性能评估 我们在 128k 上下文长度的几个基准上评估 Kimi Linear 与三个基线模型——MLA、GDN-H 和 Kimi Linear（RoPE）——的长上下文性能（见表5）。结果突出了 Kimi Linear 在这些长上下文任务中的明显优势。它始终优于 MLA 和 GDN-H，在 RULER（84.3）和 RepoQA（68.5）上取得最高分，优势显著。这种模式在大多数其他任务上也成立，除了 LongBench V2 和 Frames。总体而言，Kimi Linear 实现最高平均分数（54.5），进一步强化了其作为长上下文场景中领先注意力架构的有效性。

RL 结果 为比较 Kimi Linear 和 MLA 的 RL 收敛特性，我们使用 [50] 的内部数学训练集进行 RLVR，并在数学测试集（例如 AIME 2025、MATH500）上进行评估，同时保持算法和所有超参数相同以确保性能比较的公平性。

如图6所示，Kimi Linear 相比 MLA 表现出更好的效率。在训练集上，尽管两个模型从相似的点开始，Kimi Linear 的训练准确率增长速度明显快于 MLA，差距逐渐扩大。在测试集上，观察到类似现象。例如，在 MATH500 和 AIME2025 上，Kimi Linear 相比 MLA 实现更快更好的改进。总体而言，在 RL 下推理密集型的长形式生成中，我们凭经验观察到 Kimi Linear 表现明显优于 MLA。

总体发现总结在预训练和 SFT 阶段，建立了清晰的性能层次：Kimi Linear 优于 GDN-H，而 GDN-H 又优于 MLA。然而，在长上下文评估中，这种层次发生了变化。虽然 Kimi Linear 保持其顶级位置，

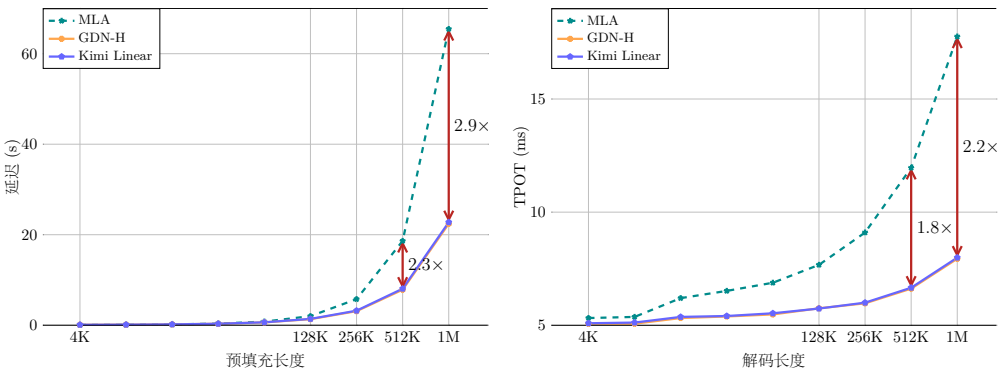


(a) (b) (c)

图 6: Kimi Linear@1.4T 和 MLA@1.4T 在数学 RL 训练期间的训练和测试准确率曲线。在整个 RL 过程中，Kimi Linear 始终以显著优势超越全注意力基线。

但 GDN-H 的性能下降，使其落后于 MLA。此外，在 RL 阶段，Kimi Linear 也表现出优于 MLA 的性能。总体而言，Kimi Linear 始终排名为顶级表现者，确立了自己作为全注意力架构优越替代品的地位。

5.6 效率比较



(a) (b)

图 7: (a) MLA（全注意力）、混合 GDN-H 和我们的 Kimi Linear 的预填充时间。(b) MLA、GDN-H 和 Kimi Linear 在解码期间的每输出令牌时间（TPOT）。（我们在测试中使用批次大小为 1。）

预填充与解码速度 我们在图7a和图7b中比较了全注意力 MLA [19]、GDN-H 和 Kimi Linear 的训练和解码时间。注意所有模型都基于 Kimi Linear 48B 设置，具有相同的层数和注意力头数。我们观察到：1）尽管引入了更细粒度的衰减机制，Kimi Linear 在预填充期间与 GDN-H 相比引入了可忽略的延迟开销。如图7a所示，它们的性能曲线几乎无法区分，确认我们的方法保持高效率。混合 Kimi Linear 模型随着序列长度增加相比 MLA 基线表现出明显的效率优势。虽然其在较短长度（4k-16k）上的性能与 MLA 相当，但从 128k 开始变得显著更快。这种效率差距在规模上急剧扩大，Kimi Linear 在 512k 序列上优于 MLA 2.3 倍，在 1M 序列上 2.9 倍。如图1b所示，Kimi Linear 在解码阶段充分展示了其优势。对于 1M 上下文长度的解码，Kimi Linear 比全注意力快 6×。

6 讨论

6.1 作为可学习位置嵌入的 Kimi Delta Attention

Transformer 中的标准注意力在设计上对其输入的序列顺序是不可知的 [99]，因此需要显式位置编码 [75, 86]。在各种方法中，RoPE [88] 由于其有效性已成为现代 LLM 中的事实标准 [98, 1, 19]。像 RoPE 这样的乘法位置编码机制可以通过广义注意力公式分析：

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

其中第 t 个查询 $\boldsymbol{q}_t$ 与第 i 个键 $\boldsymbol{k}_i$ 之间的位置关系由累积矩阵乘积反映。RoPE 将变换矩阵 $\mathbf{R}_j$ 定义为由 $d_k/2$ 个 2D 旋转矩阵 $\mathbf{R}_j^k = \begin{pmatrix} \cos(j\theta_k) & -\sin(j\theta_k) \\ \sin(j\theta_k) & \cos(j\theta_k) \end{pmatrix}$ 组成的块对角矩阵，具有**每 2 维**角频率 θ_k。由于旋转矩阵的性质，即 $\mathbf{R}_{t-i} = \mathbf{R}_t^\top \mathbf{R}_i$，绝对位置信息 $\mathbf{R}_t$ 和 $\mathbf{R}_i$ 可以分别应用于 $\boldsymbol{q}_t$ 和 $\boldsymbol{k}_i$，然后转换为相对位置信息 $t-i$，编码为 $\prod_{j=i+1}^{t} \mathbf{R}_j = \begin{pmatrix} \cos((t-i)\theta_k) & -\sin((t-i)\theta_k) \\ \sin((t-i)\theta_k) & \cos((t-i)\theta_k) \end{pmatrix}$。

因此，我们表明带有门控 delta 规则的线性注意力可以在公式12中以可比较的形式表达。其他注意力变体的类似形式总结在表6中。

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

从这个角度来看，GDN 可以被解释为一种乘法位置编码形式，其转移矩阵是数据依赖的，从而放松了 RoPE 施加的正交性约束，可能更强大 [115]。[6] 这为 RoPE 已知的外推问题提供了潜在解决方案，其固定频率可能导致对训练期间看到的上下文长度过拟合 [108, 72]。最近的一些工作采用部分 RoPE [7] 等变通方法，甚至完全放弃显式位置编码（NoPE） [49, 76, 19]。鉴于 GDN 充当类似于 RoPE 的角色，我们在模型中对全局全注意力层（MLA）选择 NoPE，允许位置信息由我们提出的 KDA 模型动态捕获。

此外，RoPE 的一个关键优势是其细粒度位置编码，通过为每对维度分配不同的旋转频率实现，其功能类似于特征维度上的非均匀傅里叶变换 [7, 41]。然而，标准 GDN 采用每头标量衰减，缺乏这种每维多样性，这促使我们提出具有可学习通道级门控的 KDA。

[6] 当保持正交性时，绝对位置编码可以独立应用于 $\boldsymbol{q}$ 和 $\boldsymbol{k}$，然后在注意力计算期间自动转换为相对位置编码 [87]。

表 6: 注意力机制在数学上等价的循环形式（$\boldsymbol{o}_t$）和并行形式（$\mathbf{O}$）的概述。为简洁起见，我们省略了归一化项和 β_t。函数 ϕ 指对应于指数核的无限维特征空间，即 $\phi(\boldsymbol{q})^\top \phi(\boldsymbol{k}) = \exp(\boldsymbol{q}^\top \boldsymbol{k})$。

	Recurrent form	Parallel form
SA [99]	$\sum_{j=1}^{t} \exp\left(\boldsymbol{q}_t^\top \boldsymbol{k}_j\right) \boldsymbol{v}_j$	$\left(\exp\left(\mathbf{QK}^\top\right) \odot \mathbf{M}\right) \mathbf{V}$
SA + RoPE [88]	$\sum_{j=1}^{t} \exp\left(\boldsymbol{q}_t^\top \left(\prod_{s=j+1}^{t} \boldsymbol{R}_s\right) \boldsymbol{k}_j\right) \boldsymbol{v}_j$	$\left(\exp\left(\mathbf{R}\left(\mathbf{Q}\right)\mathbf{R}\left(\mathbf{K}\right)^\top\right) \odot \mathbf{M}\right) \mathbf{V}$
LA [101]	$\sum_{j=1}^{t} \left(\boldsymbol{q}_t^\top \boldsymbol{k}_j\right) \boldsymbol{v}_j$	$\left(\mathbf{QK}^\top \odot \mathbf{M}\right) \mathbf{V}$
Mamba2 [16]	$\sum_{j=1}^{t} \left(\boldsymbol{q}_t^\top \left(\prod_{s=j+1}^{t} \alpha_s\right) \boldsymbol{k}_j\right) \boldsymbol{v}_j$	$\left(\mathbf{QK}^\top \odot \mathcal{A} \odot \mathbf{M}\right) \mathbf{V}$
GLA [114]	$\sum_{j=1}^{t} \left(\boldsymbol{q}_t^\top \left(\prod_{s=j+1}^{t} \mathrm{Diag}\left(\boldsymbol{\alpha}_s\right)\right) \boldsymbol{k}_j\right) \boldsymbol{v}_j$	$\left(\left(\mathbf{Q} \odot \right)\left(\overset{\mathbf{K}}{}\right)^\top \odot \mathbf{M}\right) \mathbf{V}$
DeltaNet [84]	$\sum_{j=1}^{t} \left(\boldsymbol{q}_t^\top \left(\prod_{s=j+1}^{t} \left(\mathbf{I} - \boldsymbol{k}_s \boldsymbol{k}_s^\top\right)\right) \boldsymbol{k}_j\right) \boldsymbol{v}_j$	$\left(\mathbf{QK}^\top \odot \mathbf{M}\right) \left(\mathbf{I} + \mathbf{KK}^\top \odot \mathbf{M}^-\right)^{-1} \mathbf{V}$
FoX [58]	$\sum_{j=1}^{t} \exp\left(\boldsymbol{q}_t^\top \boldsymbol{k}_j\right) \left(\prod_{s=j+1}^{t} \alpha_s\right) \boldsymbol{v}_j$	$\left(\exp\left(\mathbf{QK}^\top\right) \odot \mathcal{A} \odot \mathbf{M}\right) \mathbf{V}$
DeltaFormer [125]	$\sum_{j=1}^{t} \left(\phi(\boldsymbol{q}_t)^\top \left(\prod_{s=j+1}^{t} \left(\mathbf{I} - \phi\left(\boldsymbol{k}_s\right)\phi\left(\boldsymbol{w}_s\right)^\top\right)\right) \phi\left(\boldsymbol{k}_j\right)\right) \boldsymbol{v}_j$	$\left(\exp\left(\mathbf{QK}^\top\right) \odot \mathbf{M}\right) \left(\mathbf{I} + \exp\left(\mathbf{WK}^\top\right) \odot \mathbf{M}^-\right)^{-1} \mathbf{V}$
PaTH-FoX [115]	$\sum_{j=1}^{t} \exp\left(\boldsymbol{q}_t^\top \left(\prod_{s=j+1}^{t} \left(\mathbf{I} - \boldsymbol{w}_s \boldsymbol{w}_s^\top\right)\right) \boldsymbol{k}_j\right) \left(\prod_{s=j+1}^{t} \alpha_s\right) \boldsymbol{v}_j$	$\left(\exp\left(\left(\mathbf{QK}^\top \odot \mathbf{M}\right)\left(\mathbf{I} + \mathbf{WW}^\top \odot \mathbf{M}^-\right)^{-1}\right) \odot \mathcal{A} \odot \mathbf{M}\right) \mathbf{V}$
GDN [111]	$\sum_{j=1}^{t} \left(\boldsymbol{q}_t^\top \left(\prod_{s=j+1}^{t} \alpha_s \left(\mathbf{I} - \boldsymbol{k}_s \boldsymbol{k}_s^\top\right)\right) \boldsymbol{k}_j\right) \boldsymbol{v}_j$	$\left(\mathbf{QK}^\top \odot \mathcal{A} \odot \mathbf{M}\right) \left(\mathbf{I} + \mathbf{KK}^\top \odot \mathcal{A} \odot \mathbf{M}^-\right)^{-1} \mathbf{V}$
Comba [40]	$\sum_{j=1}^{t} \left(\boldsymbol{q}_t^\top \left(\prod_{s=j+1}^{t} \left(\alpha_s - \boldsymbol{k}_s \boldsymbol{k}_s^\top\right)\right) \boldsymbol{k}_j\right) \boldsymbol{v}_j$	$\left(\mathbf{QK}^\top \odot \mathcal{A} \odot \mathbf{M}\right) \left(\mathbf{I} + \mathbf{KK}^\top \odot \mathcal{A}^{i-1/j} \odot \mathbf{M}^-\right)^{-1} \mathbf{V}$
RWKV7 [71]	$\sum_{j=1}^{t} \left(\boldsymbol{q}_t^\top \left(\prod_{s=j+1}^{t} \left(\mathrm{Diag}\left(\boldsymbol{\alpha}_s\right) - \left(\boldsymbol{b}_s \odot \hat{\boldsymbol{k}}_s\right) \hat{\boldsymbol{k}}_s^\top\right)\right) \boldsymbol{k}_j\right) \boldsymbol{v}_j$	$\left(\left(\mathbf{Q} \odot \right)\left(\overset{\mathbf{K}}{}\right)^\top \odot \mathbf{M}\right) \left(\mathbf{I} + \left(\hat{\mathbf{K}} \odot {}^{0\to t-1}\right)\left(\overset{\hat{\mathbf{K}} \odot \mathbf{B}}{}\right)^\top \odot \mathbf{M}^{-1}\right)^{-1} \mathbf{V}$
KDA (ours)	$\sum_{j=1}^{t} \left(\boldsymbol{q}_t^\top \left(\prod_{s=j+1}^{t} \mathrm{Diag}\left(\boldsymbol{\alpha}_s\right)\left(\mathbf{I} - \boldsymbol{k}_s \boldsymbol{k}_s^\top\right)\right) \boldsymbol{k}_j\right) \boldsymbol{v}_j$	$\left(\left(\mathbf{Q} \odot \right)\left(\overset{\mathbf{K}}{}\right)^\top \odot \mathbf{M}\right) \left(\mathbf{I} + \left(\mathbf{K} \odot \right)\left(\overset{\mathbf{K}}{}\right)^\top \odot \mathbf{M}^{-1}\right)^{-1} \mathbf{V}$

6.2 与 DPLR 的关系

（门控）DeltaNet 可以推广到更具表现力的对角加低秩（DPLR）结构，定义为 $\mathbf{D} - \boldsymbol{a}_t \boldsymbol{b}_t^\top$。这种结构也在 S4 等模型中探索 [30]，它采用静态 DPLR 公式作为状态转移矩阵。在计算期间，该矩阵通常被联合对角化到复平面，从而将其表现力限制为对角变换 [64]。

虽然 DPLR 结构引入了更丰富的模型交互，并可能通过其键值更新规则增强回忆，但它也存在一个显著限制：高计算成本和较差的可并行性。这些缺点使 DPLR 在大规模或实时场景中本质上更慢，其中保持参数效率成为关键设计挑战。

为解决此问题，KDA 引入了 DPLR 的约束变体，其中公式1可以重写为 $\mathbf{S}_t = \left(\mathrm{Diag}(\boldsymbol{\alpha}_t) - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top \mathrm{Diag}(\boldsymbol{\alpha}_t)\right) \mathbf{S}_{t-1} + \beta_t \boldsymbol{k}_t \boldsymbol{v}_t^\top$ 两者之间的对应关系为：

$$\mathbf{S}_t = (\mathbf{D} - \boldsymbol{a}_t \boldsymbol{b}_t^\top)\mathbf{S}_{t-1} + \boldsymbol{k}_t \boldsymbol{v}_t^\top, \text{s.t.,} \quad \mathbf{D} = \mathrm{Diag}(\boldsymbol{\alpha}_t), \boldsymbol{a}_t = \beta_t \boldsymbol{k}_t, \boldsymbol{b}_t = \boldsymbol{k}_t \odot \boldsymbol{\alpha}_t.$$

此外，通过共享 $\boldsymbol{\alpha}_t$，我们可以像公式1中那样将其分解出来，实现对 $\mathbf{S}_t$ 的细粒度乘法衰减，方式类似于 GLA [114]，然后像 DeltaNet [84, 112] 那样进行 Householder 风格变换以高效更新状态。我们在清单8a和清单8b中提供了 DPLR 和 KDA 的分块 PyTorch 风格伪代码实现的并排比较。关键改进如下：

- 清单8a 第 13-16 行 vs. 清单8b 第 14-15 行：分块形式中累积衰减项 $1/\Gamma$ 的倒数（公式9）可能引入数值不稳定性。虽然我们可以通过二级分块 [113] 解决此问题，但它会产生额外的计算和 I/O 开

```
def chunk_dplr(q, k, v, a, b, g, chunk_size):
    B, H, T, K, V, BT = *q.shape, v.shape[-1], chunk_size
    NT, S = T // BT, k.new_zeros(B, H, K, V)
    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])
    gc = g.cumsum(-2)
-   Aab, Aak, Aqb, Aqk = (torch.zeros(B, H, NT, BT, BT) for _ in
 ↪  range(4))

    for i in range(BT):
        a_i, q_i, g_i = (x[:,:,:,i,None] for x in (a, q, gc))
        mask = (torch.arange(BT) <= i)[..., None]
        s1_i = (g_i - gc).exp().where(mask, 0)
        s2_i= (g_i - g[:,:,:,i,None] - gc).where(mask, 0)
-       Aqk[..., i, :] = (q_i * k * s1_i).sum(-1)
-       Aqb[..., i, :] = (q_i * b * s1_i).sum(-1)
-       Aab[..., i, :] = (a_i * b * s2_i).sum(-1)
-       Aak[..., i, :] = (a_i * k * s2_i).sum(-1)
    for i in range(1, BT):
        Aab[..., i, :i] = Aab[..., i, :i] + (Aab[..., i, :, None]
       ↪  * Aab[..., :, :i]).sum(-2)
    Aab = Aab + torch.eye(BT)
    u, w = Aab @ (Aak @ v), Aab @ ((gc-g).exp() * a)
    o = torch.zeros_like(v)
    mask = torch.triu(torch.ones(BT, BT), diagonal=1)
    for i in range(0, NT):
        q_i, k_i, v_i, u_i, w_i, b_i = (x[:, :, i] for x in (q,
       ↪  k, v, u, w, b))
-       o1 = Aqk[:, :, i] @ v_i
-       o2 = Aqb[:, :, i] @ (u_i + w_i @ S)
-       o3 = (q_i * gkc[:, :, i].exp()) @ S
        o[:, :, i] = o1 + o2 + o3
        decay = (gc[:, :, i, -1, None] - gc[:, :, i]).exp()
        S = S * gc[:,:,i,-1,:,None].exp()
-       S +=(k_i * decay).transpose(-1,-2) @ v_i
-       S +=(b_i * decay).transpose(-1,-2) @ (u_i + w_i @ S)
    return o, S
```

(a) 分块 DPLR 的 PyTorch 风格伪代码。

```
def chunk_kda(q, k, v, a, b, g, chunk_size):
    B, H, T, K, V, BT = *q.shape, v.shape[-1], chunk_size
    NT, S = T // BT, k.new_zeros(B, H, K, V)
    q, k, v, g = map(lambda x: rearrange(x, 'b h (n c) ... -> b h
    ↪  n c ...', c=BT), [q, k, v, g])
    gc = g.cumsum(-2)
+   Aqk, Akk = (torch.zeros(B, H, NT, BT, BT) for _ in range(2))

    for i in range(BT):
        k_i, q_i = k[:, :, :, i, None], q[:, :, :, i, None]
        g_i = gc[...,i:i+1,:]
        mask = (torch.arange(BT) <= i)[..., None]
        s1_i = (g_i - gc).exp().where(mask, 0)
        s2_i = (gc - g_i).exp()
+       Aqk[:, :, :, i, :] = (q_i * k * s1_i).sum(-1)
+       Akk[..., i] = (k_i * k * s2_i).sum(-1)
    mask = torch.triu(torch.ones(BT, BT), diagonal=0)
    A = -Akk.masked_fill(mask, 0)
    for i in range(1, BT):
        A[..., i, :i] = A[..., i, :i] + (A[..., i, :, None] *
       ↪  A[..., :, :i].clone()).sum(-2)
    A = (A + torch.eye(BT))
    w, u = A @ (gc.exp() * k), A @ v
    o = torch.zeros_like(v)
    mask = torch.triu(torch.ones(BT, BT), diagonal=1)
    for i in range(0, NT):
        q_i, k_i, u_i, g_i, w_i = (x[:, :, i] for x in (q, k, u,
       ↪  gc, w))
+       o[:,:,i]=(q_i *g_i.exp()) @ S + Aqk @(u_i-w_i @ S)
        decay = (g_i[:,:,-1:] - g_i).exp()
        S = S * g_i[:, :, -1, :, None].exp()
+       S += (k_i * decay).transpose(-1,-2) @ v_i
    return o, S
```

(b) 分块 KDA 的 PyTorch 风格伪代码。

销。通过在 DPLR 公式中固定 $\boldsymbol{a} = \boldsymbol{b} = \boldsymbol{k}$，KDA 消除了两个二级分块步骤的需要，大幅减少了冗余操作并提高了整体效率。

- 清单8a 第 25-27,31-32 行 vs. 清单8b 第 26,29 行：KDA 进一步消除了块间和输出计算期间大约三个矩阵乘法，导致显著的核级加速。

我们进一步在图2中对核速度进行基准测试，显示 KDA 在长达 64k 的序列长度上实现接近 DPLR 2× 的速度。

6.3 复杂度分析

训练 FLOPs 我们在 Kimi Linear 中保持与全注意力 MLA 相似的参数数量。线性投影计算与全局注意力层保持相同。关键区别在于与注意力计算相关的 FLOPs。为简单起见，我们专注于非变长场景。基于门控规则核的实现，固定块大小 $C = 64$ 和头维度 d_h 的门控 delta 规则 [102]（每序列长度 T）的

理论 FLOPs 如下：

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

对于全（全局）注意力，每头的主导项为

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

推理策略与成本 Kimi Linear 中的推理策略采用混合方法以优化计算和 I/O 效率。在预填充阶段，模型使用计算密集型块核（见 § 3.1），而在自回归生成期间切换到更高效的循环核（公式2）。线性 KDA 的一个关键优势是它能够保持固定大小的状态（每头 $d_k \times d_v$，$d_k = d_v = 128$），无论序列长度如何。对于我们的混合模型，随着序列长度增加，I/O 受限的解码时间接近最大混合效率比 3:1 相比全注意力。这一趋势反映在图7b中，Kimi Linear 在 100 万令牌上下文处实现 2.3× 加速。此外，通过消除对大型线性缩放 KV 缓存的需求，Kimi Linear 能够重新分配内存资源以支持更大的批次大小，提高整体吞吐量。在长上下文场景（长达 100 万令牌）中，这种内存效率导致理论解码加速高达 6.3×（见图1b）。

7 相关工作

7.1 高效次二次注意力

标准自注意力机制的二次时间复杂度 [99] 仍然是基于 Transformer 模型处理长上下文的基本瓶颈。随着大型语言模型（LLM）现在期望处理百万令牌序列以完成智能体工具使用和仓库级代码分析等任务，这一限制变得越来越关键 [19, 50]。为克服这一挑战，大量研究探索了更高效的注意力机制 [91, 89]，大致可分为两个主要方向：（1）线性注意力，和（2）稀疏注意力。

线性注意力 将二次注意力图重新表述为核化特征交互，用正特征映射替换 softmax，使注意力可以通过两个关联矩阵乘积计算 [48]。这消除了显式的 $\mathcal{O}(T^2)$ 相似性矩阵，实现相对于序列长度的线性时间计算。后续工作通过更精细的记忆控制显著增强了普通线性注意力，从数据独立的 “衰减” [92, 78] 转向更具适应性的数据依赖机制 [29, 93]，并将衰减粒度从粗略的头级 [16] 细化为精确的通道级衰减。GLA 用对角通道级门控概括了这些方法，在保持分块并行性的同时平衡表现力和效率 [113, 114]。表7总结了相应的更新规则。总体而言，这些方法将注意力视为用并行前缀扫描算子和融合矩阵乘法更新的紧凑循环记忆，与现代加速器对齐良好 [42]。

一个互补的观点将线性注意力连接到快速权重记忆 [84]：状态是由类 Hebbian 规则在线更新的低容量关联表 [69]，而慢权重分摊何时存储、更新或遗忘 [68]。

在表7中，我们提供了现有高效令牌混合方法的总结，从状态更新机制和最优化目标的角度进行比较。

从这个角度来看，门控和衰减作为可学习标准，减轻干扰并稳定优化 [90]。尽管有这些进展，线性注意力在极端长上下文检索的精确复制和细粒度选择方面仍然落后于全注意力。这促使了混合设计（交错线性和全注意力）和更结构化更新的发展。特别是，GDN/KDA 使用的门控 delta 规则向快速权重状态引入秩-1 校正更新，改善目标保留，同时在算子级别保持可并行性 [112]。

带门控机制的线性注意力 普通线性注意力 [48] 已知缺乏 softmax 注意力固有的选择机制 [99]，在表现力方面不足。为解决此问题，门控线性注意力模型作为内存高效且可并行的替代方案出现 [113, 114,

表 7: 通过状态更新规则及其在 TTT 框架下学习目标的视角对不同注意力机制的概述 [90]。为简洁起见，我们忽略所有归一化项和激活/核函数。

	Objective $\mathcal{L}$	Update rule $\mathbf{S}_t = \mathbf{S}_{t-1} - \nabla_{\mathbf{S}_{t-1}}\mathcal{L}$
LA [48]	$-\langle \mathbf{S}_{t-1}^\top \boldsymbol{k}_t, \boldsymbol{v}_t \rangle$	$\mathbf{S}_t = \mathbf{S}_{t-1} + \boldsymbol{k}_t \boldsymbol{v}_t^\top$
RetNet [92]	$-\beta_t \langle \mathbf{S}_{t-1}^\top \boldsymbol{k}_t, \boldsymbol{v}_t \rangle + \frac{1}{2} \left\| \sqrt{1-\alpha}\, \mathbf{S}_{t-1} \right\|_F^2$	$\mathbf{S}_t = \alpha \mathbf{S}_{t-1} + \beta_t \boldsymbol{k}_t \boldsymbol{v}_t^\top$
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 [114]	$-\langle \mathbf{S}_{t-1}^\top \boldsymbol{k}_t, \boldsymbol{v}_t \rangle + \frac{1}{2} \left\| \sqrt{\mathrm{Diag}\,(\boldsymbol{1}-\boldsymbol{\alpha}_t)} \mathbf{S}_{t-1} \right\|_F^2$	$\mathbf{S}_t = \mathrm{Diag}(\boldsymbol{\alpha}_t) \mathbf{S}_{t-1} + \boldsymbol{k}_t \boldsymbol{v}_t^\top$
HGRN2 [77]	$-\langle \mathbf{S}_{t-1}^\top (\boldsymbol{1}-\boldsymbol{\alpha}_t), \boldsymbol{v}_t \rangle + \frac{1}{2} \left\| \sqrt{\mathrm{Diag}\,(\boldsymbol{1}-\boldsymbol{\alpha}_t)} \mathbf{S}_{t-1} \right\|_F^2$	$\mathbf{S}_t = \mathrm{Diag}(\boldsymbol{\alpha}_t) \mathbf{S}_{t-1} + (\boldsymbol{1}-\boldsymbol{\alpha}_t) \boldsymbol{v}_t^\top$
Longhorn [59]	$\frac{1}{2} \left\| \mathbf{S}_{t-1}^\top \boldsymbol{k}_t - \boldsymbol{v}_t \right\|_{\mathrm{Diag}(\boldsymbol{\beta}_t)}^2$	$\mathbf{S}_t = \left(\mathbf{I} - \frac{\boldsymbol{\beta}_t}{1+\boldsymbol{\beta}_t \boldsymbol{k}_t^\top \boldsymbol{k}_t} \boldsymbol{k}_t \boldsymbol{k}_t^\top \right) \mathbf{S}_{t-1} + \beta_t \boldsymbol{k}_t \boldsymbol{v}_t^\top$
Comba [40]	$\frac{\beta_t}{2} \left\| \mathbf{S}_{t-1}^\top \boldsymbol{k}_t - \boldsymbol{v}_t \right\|^2 + \frac{1}{2} \left\| \sqrt{1-\alpha_t} \mathbf{S}_{t-1} \right\|_F^2$	$\mathbf{S}_t = \left(\alpha_t - \beta_t \boldsymbol{k}_t \hat{\boldsymbol{k}}_t^\top \right) \mathbf{S}_{t-1} + \beta_t \boldsymbol{k}_t \boldsymbol{v}_t^\top$
RWKV7 [71]	$\frac{1}{2} \left\| \mathbf{S}_{t-1}^\top \tilde{\boldsymbol{k}}_t - \boldsymbol{v}_t \right\|^2 + \frac{1}{2} \left\| \sqrt{\mathrm{Diag}\,(\boldsymbol{1}-\boldsymbol{\alpha}_t)} \mathbf{S}_{t-1} \right\|_F^2$	$\mathbf{S}_t = \left(\mathrm{Diag}\,(\boldsymbol{\alpha}_t) - \left(\boldsymbol{b}_s \odot \hat{\boldsymbol{k}}_s \right) \hat{\boldsymbol{k}}_t^\top \right) \mathbf{S}_{t-1} + \boldsymbol{k}_t \boldsymbol{v}_t^\top$
GDN [111]	$\frac{\beta_t}{2} \left\| \tilde{\mathbf{S}}_{t-1}^\top \boldsymbol{k}_t - \boldsymbol{v}_t \right\|^2$	$\mathbf{S}_t = \left(\mathbf{I} - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top \right) \alpha_t \mathbf{S}_{t-1} + \beta_t \boldsymbol{k}_t \boldsymbol{v}_t^\top$
KDA (ours)	$\frac{\beta_t}{2} \left\| \tilde{\mathbf{S}}_{t-1}^\top \boldsymbol{k}_t - \boldsymbol{v}_t \right\|^2$	$\mathbf{S}_t = \left(\mathbf{I} - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top \right) \mathrm{Diag}\,(\boldsymbol{\alpha}_t) \mathbf{S}_{t-1} + \beta_t \boldsymbol{k}_t \boldsymbol{v}_t^\top$

对于 GDN 和 KDA，更新可以看作是在衰减状态 $\tilde{\mathbf{S}}$ 上执行随机梯度下降（SGD）过程，即 $\mathbf{S}_t = \tilde{\mathbf{S}}_{t-1} - \nabla_{\tilde{\mathbf{S}}_{t-1}}\mathcal{L}$，其中 $\tilde{\mathbf{S}}_{t-1}$ 由标量或细粒度门控衰减。

29]。这些模型不存储不断扩展的 KV 缓存，而是采用固定大小的矩阵值状态和可学习门控来有选择地保留和遗忘信息。这种设计实现与 softmax 注意力相当的表现力 [65, 125, 64]，同时在推理期间保持恒定的时间和内存复杂度。这种模型记忆更新 $\mathbf{S}_t \in \mathbb{R}^{d_k \times d_v}$ 的一般循环公式可以表示为：

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

各种门控线性注意力机制之间的主要区别在于遗忘门 $\mathbf{A}_t$ 的参数化，如表7所总结。例如，RetNet [92] 使用数据独立标量衰减 α，Mamba2 [16] 采用数据依赖标量 α_t。具体而言，GLA [114] 利用对角化细粒度矩阵 $\mathrm{Diag}(\boldsymbol{\alpha}_t) \in \mathbb{R}^{d_k \times d_k}$，在效率和性能之间提供有效权衡。其他变体显示在表7中。

稀疏注意力 大量工作通过利用其固有的稀疏性来降低标准注意力的二次复杂度，通过在策略选择的令牌子集上执行计算来近似全注意力分数。核心挑战在于有效识别该子集而不降低模型性能。早期方法通常利用高效的免训练静态模式，如滑动和扩张窗口 [20, 31, 107] 或固定模式 [120, 35]，但其刚性结构通常会损害模型准确性。更先进的方法基于上下文确定重要位置，如聚类 [51, 106] 和轻量级路由机制 [25, 73, 2, 9]，但这种动态选择过程引入计算开销，可能阻止它们在没有专用核加速的情况下实现其全部理论加速 [21]。一些模型进一步在推理阶段引入免训练稀疏化 [107, 109]。

最近稀疏注意力的方法开始优先考虑硬件协同设计，如 NSA [119, 96] 和 MoBA [63] 所例证，两者都从令牌级选择转向块级选择。在 NSA 中，每个查询基于 MLP 产生的分数动态选择块。该方法的效率依赖于其对分组查询注意力（GQA） [98] 的使用，具有大头数（通常是 16 的倍数），这种配置专门设计用于通过高度并行的张量-矩阵乘法加速计算。类似地，MoBA 执行 top-k 块选择，但利用通过 flash-attention 核 [17] 高效计算的对数和指数（LSE）分数。与 NSA 和 MoBA 相比，最近提出的

DeepSeek-V3.2-Exp Attention（DSA） [18] 恢复了令牌级稀疏性，通过使用低精度 fp8 实现的可学习全注意力索引器和用于令牌选择的小头维度来保持效率。

讨论 线性注意力和稀疏注意力代表了高效长上下文建模的两条不同路径。稀疏注意力倾向于更有效地检索细粒度历史信息，但这种优势以存储整个 KV 缓存用于令牌选择为代价，使其不如保持恒定状态的线性注意力模型高效。此外，稀疏注意力仅执行信息选择，其理论表现上限仍然是全注意力。相比之下，基于“压缩即智能”原则的线性注意力，通过固定大小状态实现泛化，当与 Delta 学习规则结合时，可以实现理论上更强的表现能力。尽管线性注意力传统上因检索能力弱而受到批评，但这种限制可以通过状态扩展 [23, 34, 117, 39] 或相关技术来缓解。尽管如此，尽管有这些优势，线性注意力仍然受到当前硬件实现和缺乏优化推理基础设施的限制。我们的工作通过 Kimi Linear 克服了这些限制，这是一个与 vLLM 集成以实现高效推理的强大模型。我们提出的 KDA 与全注意力基线相比具有竞争力的性能（表3），在百万令牌上下文处实现超过 2× 的解码加速（图7b）。尽管它们对高效长上下文建模的方法不同，线性注意力和稀疏注意力并非互斥。未来工作可以探索整合两者优势的混合模型，利用线性注意力的压缩和泛化能力与稀疏注意力的细粒度检索优势，以进一步提高模型性能和效率。

7.2 混合模型

尽管效率高，纯线性注意力仍然在精确记忆检索和精确复制方面挣扎 [45, 104] 这种缺陷阻碍了它们在工业规模 LLM 中的采用，其中稳健的长上下文回忆（例如超过 100 万令牌）和跨广泛代码仓库的可靠工具使用至关重要 [50]。最近的工作表明，线性注意力和全注意力可以有效互补，导致各种混合设计。

层内混合 一类混合架构是层内混合，它在每层内自适应地融合不同机制的输出。一种常见实现在每层内融合来自异构头的输出，如结合标准注意力与状态空间模型（SSM） [22, 56]。相比之下，序列级方法将不同机制应用于输入的不同部分。例如，一些对过去上下文使用线性注意力，对最近令牌使用 SWA [123, 54, 67]，而 NHA [24] 用 GSA [124] 压缩历史并将其与局部滑动窗口上下文结合以模拟标准注意力操作。

层间混合 层内混合的一个关键缺点是系统复杂性增加和推理开销。异构机制需要单独的计算路径，使分布式并行等优化复杂化。为缓解这一挑战，层间混合已成为 LLM 中更广泛采用和实用的策略 [66, 57, 97]。这种方法涉及以预定义比例堆叠不同类型的层，如全注意力和线性替代方案。基于这一范式，我们实现了一个简单而有效的策略：以固定 3:1 比例交错线性和全注意力层（见 § 5.2进行消融）。这种规则重复的结构简化了 KV 缓存管理，并无缝集成标准优化。对于混合的线性组件，我们偏离使用 Mamba2 [16] 的常见做法。相反，我们采用 KDA，因为我们发现它产生更优越的整体性能，特别是在检索和复制能力方面。

讨论 最近的工作表明，混合模型可能对 RoPE 基频调整敏感，这种脆弱性使上下文窗口扩展复杂化 [126]。这种敏感性可能阻碍模型外推到更长序列的能力。为解决这一挑战，最近模型趋向于纳入无位置编码（NoPE）的解决方案。例如，Falcon-H [126] 使用非传统的高基频（例如 $b \approx 10^{11}$）将其位置编码推向接近 NoPE 状态。在架构上，SwanGPT [76] 将基于 RoPE 的层与基于 NoPE 的全注意力层交错。与这一方向一致，我们发现将 KDA 层与 NoPE 全注意力混合也是一种非常有效的策略，促进直接的上下文窗口扩展。

结论

我们介绍了 Kimi Linear，一种混合线性注意力架构，旨在满足智能体智能和测试时扩展的效率需求，同时不牺牲质量。Kimi Linear 的核心是 Kimi Delta Attention（KDA），这是一种具有通道级门控机制的先进线性注意力模块，可增强记忆控制并使 RNN 风格模型在混合架构中发挥作用。通过以 3:1 比例将 KDA 与全局注意力交错，Kimi Linear 将内存使用量减少多达 75%，同时实现高达 6.3× 的更高解码吞吐量，并超越全注意力基线。我们的方法为大型语言模型提供了可扩展、高效的解决方案，开源 KDA 内核和预训练检查点促进了进一步的研究。

参考文献

[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 multitasks”. 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 NeurIPS* 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 NeurIPS*. 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] Albert Gu, Karan Goel, and Christopher Ré. *Efficiently Modeling Long Sequences with Structured State Spaces*. 2022. arXiv: 2111.00396 [cs.LG].

[31] 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.

[32] 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.

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

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

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

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

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

[38] 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).

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

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

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

[42] 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.

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

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

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

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

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

[48] 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.

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

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

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

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

[53] 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.

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

[55] 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/.

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

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

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

[59] 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.

[60] 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 NeurIPS*. 2023. URL: https://openreview.net/forum?id=1qvx610Cu7.

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

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

[63] 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.

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

[65] 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/.

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

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

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

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

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

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

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

[73] 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).

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

[75] 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.

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

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

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

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

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

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

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

[83] 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.

[84] 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.

[85] 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.

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

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

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

[89] 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.

[90] 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.

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

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

[93] 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.

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

[95] 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.

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

[97] 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).

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

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

[100] 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.

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

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

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

[104] 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).

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

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

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

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

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

[110] 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.

[111] 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.

[112] 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.

[113] 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.

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

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

[116] 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.

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

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

[119] 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.

[120] Manzil Zaheer et al. “Big bird: Transformers for longer sequences”. In: *Advances in NeurIPS* 33 (2020), pp. 17283–17297.

[121] 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.

[122] Biao Zhang and Rico Sennrich. “Root mean square layer normalization”. In: *Advances in NeurIPS* 32 (2019).

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

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

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

[126] 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 贡献

作者按贡献重要性排序，项目领导角色的人员排在最后。该项目在 Moonshot AI 开发，几位外部合作者标记为 #。标有星号 (*) 的名称表示已不在我们团队的人员。

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
Yuxin Wu
Xinyu Zhou
Zhilin Yang
Yulun Du

[1] 中国苏州大学

[2] 香港科技大学（广州）

[3] 麻省理工学院

[4] 中国科学院杭州医学研究所

B KDA 分块并行的推导

我们首先回顾 KDA 的循环形式：

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

我们的目标是将 $\mathbf{P}_{[t]}^r$ 和 $\mathbf{H}_{[t]}^r$ 转换为适合并行计算的矩阵形式。

我们表明，$\mathbf{P}_{[t]}^r$ 涉及广义 Householder 矩阵的累积乘积，可以使用经典 WY 表示进行优化。

命题 1. *矩阵 $\mathbf{P}_{[t]}^r$ 可以表示为：*

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

其中辅助向量 $\boldsymbol{w}_{[t]}^r \in \mathbb{R}^{d_k}$ 通过以下递推关系计算：

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

证明. 我们用数学归纳法进行证明。

归纳步骤：假设命题对 $r-1$ 成立，即 $\mathbf{P}_{[t]}^{r-1} = \operatorname{Diag}(\boldsymbol{\gamma}_{[t]}^{r-1}) - \sum_{i=1}^{r-1}\operatorname{Diag}(\boldsymbol{\gamma}_{[t]}^{i\to r-1})\boldsymbol{k}_{[t]}^i \boldsymbol{w}_{[t]}^{i\top}$。我们现在推导：

$$\begin{aligned}\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}\end{aligned}$$

归纳步骤成立。 ■

类似于 $\mathbf{P}_{[t]}^r$，$\mathbf{H}_{[t]}^r$ 也可以以可并行化的形式表达。

命题 2. *矩阵 $\mathbf{H}_{[t]}^r$ 可以表示为：*

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

其中辅助向量 $\boldsymbol{u}_{[t]}^r \in \mathbb{R}^{d_v}$ 通过以下递推关系计算：

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

证明. 我们再次使用数学归纳法。

归纳步骤：假设命题对 $r-1$ 成立。

$$\begin{aligned}
\mathbf{H}_{[t]}^r &= \left(\mathbf{I} - \beta_{[t]}^r \boldsymbol{k}_{[t]}^r \boldsymbol{k}_{[t]}^{r\top}\right) \mathrm{Diag}(\boldsymbol{\alpha}_{[t]}^r) \mathbf{H}_{[t]}^{r-1} + \beta_{[t]}^r \boldsymbol{k}_{[t]}^r \boldsymbol{v}_{[t]}^{r\top} \\
&= \left(\mathbf{I} - \beta_{[t]}^r \boldsymbol{k}_{[t]}^r \boldsymbol{k}_{[t]}^{r\top}\right) \mathrm{Diag}(\boldsymbol{\alpha}_{[t]}^r) \left(\sum_{i=1}^{r-1} \mathrm{Diag}\left(\boldsymbol{\gamma}_{[t]}^{i\to r-1}\right) \boldsymbol{k}_{[t]}^i \boldsymbol{u}_{[t]}^{i\top} \right) + \beta_{[t]}^r \boldsymbol{k}_{[t]}^r \boldsymbol{v}_{[t]}^{r\top} \\
&= \left(\mathbf{I} - \beta_{[t]}^r \boldsymbol{k}_{[t]}^r \boldsymbol{k}_{[t]}^{r\top}\right) \left(\sum_{i=1}^{r-1} \mathrm{Diag}\left(\boldsymbol{\gamma}_{[t]}^{i\to r}\right) \boldsymbol{k}_{[t]}^i \boldsymbol{u}_{[t]}^{i\top} \right) + \beta_{[t]}^r \boldsymbol{k}_{[t]}^r \boldsymbol{v}_{[t]}^{r\top} \\
&= \sum_{i=1}^{r-1} \mathrm{Diag}\left(\boldsymbol{\gamma}_{[t]}^{i\to r}\right) \boldsymbol{k}_{[t]}^i \boldsymbol{u}_{[t]}^{i\top} - \beta_{[t]}^r \boldsymbol{k}_{[t]}^r \boldsymbol{k}_{[t]}^{r\top} \sum_{i=1}^{r-1} \mathrm{Diag}\left(\boldsymbol{\gamma}_{[t]}^{i\to r}\right) \boldsymbol{k}_{[t]}^i \boldsymbol{u}_{[t]}^{i\top} + \beta_{[t]}^r \boldsymbol{k}_{[t]}^r \boldsymbol{v}_{[t]}^{r\top} \\
&= \sum_{i=1}^{r-1} \mathrm{Diag}\left(\boldsymbol{\gamma}_{[t]}^{i\to r}\right) \boldsymbol{k}_{[t]}^i \boldsymbol{u}_{[t]}^{i\top} - \boldsymbol{k}_{[t]}^r \left(\beta_{[t]}^r \sum_{i=1}^{r-1} \left(\boldsymbol{k}_{[t]}^{r\top} \mathrm{Diag}\left(\boldsymbol{\gamma}_{[t]}^{i\to r}\right) \boldsymbol{k}_{[t]}^i \right) \boldsymbol{u}_{[t]}^i \right)^\top + \boldsymbol{k}_{[t]}^r \beta_{[t]}^r \boldsymbol{v}_{[t]}^{r\top} \\
&= \sum_{i=1}^{r-1} \mathrm{Diag}\left(\boldsymbol{\gamma}_{[t]}^{i\to r}\right) \boldsymbol{k}_{[t]}^i \boldsymbol{u}_{[t]}^{i\top} + \boldsymbol{k}_{[t]}^r \left(\underbrace{\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)}_{\boldsymbol{u}_{[t]}^r} \right)^\top \\
&= \sum_{i=1}^{r-1} \mathrm{Diag}\left(\boldsymbol{\gamma}_{[t]}^{i\to r}\right) \boldsymbol{k}_{[t]}^i \boldsymbol{u}_{[t]}^{i\top} + \boldsymbol{k}_{[t]}^r \boldsymbol{u}_{[t]}^{r\top} \\
&= \sum_{i=1}^{r} \mathrm{Diag}\left(\boldsymbol{\gamma}_{[t]}^{i\to r}\right) \boldsymbol{k}_{[t]}^i \boldsymbol{u}_{[t]}^{i\top}
\end{aligned}$$

归纳步骤成立。 ■

C Pseudo Code for chunkwise KDA

```
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(), 'bhk -> bhk1')
        S += rearrange((g_i[:, :, -1:] - g_i).exp() * k_i, 'bhck -> bhkc') @ v_i
    return rearrange(o, 'b h n c d -> b (n c) h d').to(dtype)
```

Listing 1: KDA 分块形式的 PyTorch 风格伪代码片段。

D Kimi Linear@5.7T 结果

遵循 Moonlight，我们还使用扩展的 5.7T 令牌数据集训练了 Kimi Linear 以证明其有效性。凭借 3× 稀疏度和新的注意力架构设计，Kimi Linear 在几乎所有基准上都始终优于 Moonlight，凸显了新架构的功效。结果如表8所示为基础模型，表9所示为指令微调模型。Moonlight-Instruct 未在其 8K 上下文限制之外的任务上进行评估（“-”）。

Kimi Linear@5.7T 在 1M 上下文长度的 RULER 上获得 94.8 分。这种长上下文性能强化了 Kimi Linear 是全注意力架构的有前途替代品，在潜在提供更高效的资源利用的同时提供相当或更优的结果。

表 8: Kimi-Linear-Base 和 Moonlight-Base 在各种任务上的性能。

	基准	# 样本	Kimi-Linear-Base	Moonlight-Base
	架构	-	MoE	MoE
	# 激活参数	-	3B	3B
	# 总参数	-	48B	16B
	训练令牌	-	5.7T	5.7T
通用	TriviaQA	5 样本	75.2	66.2
	SimpleQA	5 样本	10.1	5.6
	MMLU-Pro	5 样本	54.8	42.4
	MMLU-redux	5 样本	79.7	73.8
	WinoGrande	5 样本	81.5	74.6
	GPQA-Diamond (avg@8)	5 样本	40.4	35.2
数学	MATH	4 样本	58.5	45.3
	GSM8k	8 样本	86.3	77.2
	GSM8k-platinum	8 样本	89.6	79.4
	CMATH	6 样本	85.5	79.6
代码	CRUXEval-I-cot	0 样本	61.0	45.9
	CRUXEval-O-cot	0 样本	67.0	46.6
	LiveCodeBench (v6)	1 样本	20.0	14.3
	EvalPlus	-	64.9	50.3
中文	C-Eval	5 样本	83.3	77.6
	CSimpleQA	5 样本	53.5	34.7

表 9: Kimi-Linear-Instruct 和 Moonlight-Instruct 在各种任务上的性能。

	基准	Kimi-Linear-Instruct	Moonlight-Instruct
	架构	MoE	MoE
	# 激活参数	3B	3B
	# 总参数	48B	16B
	训练令牌	5.7T	5.7T
通用	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
数学	AIME 2025 (Avg@64)	58.6	-
	MATH500 (准确率)	94.6	58.0
	HMMT 2025 (Avg@32)	44.5	-
代码	LiveCodeBench v6 (Pass@1)	45.7	11.9
	OJBench (Pass@1)	14.2	-
	Humaneval^{+}	70.9	46.3
	MBPP^{+}	72.4	56.3