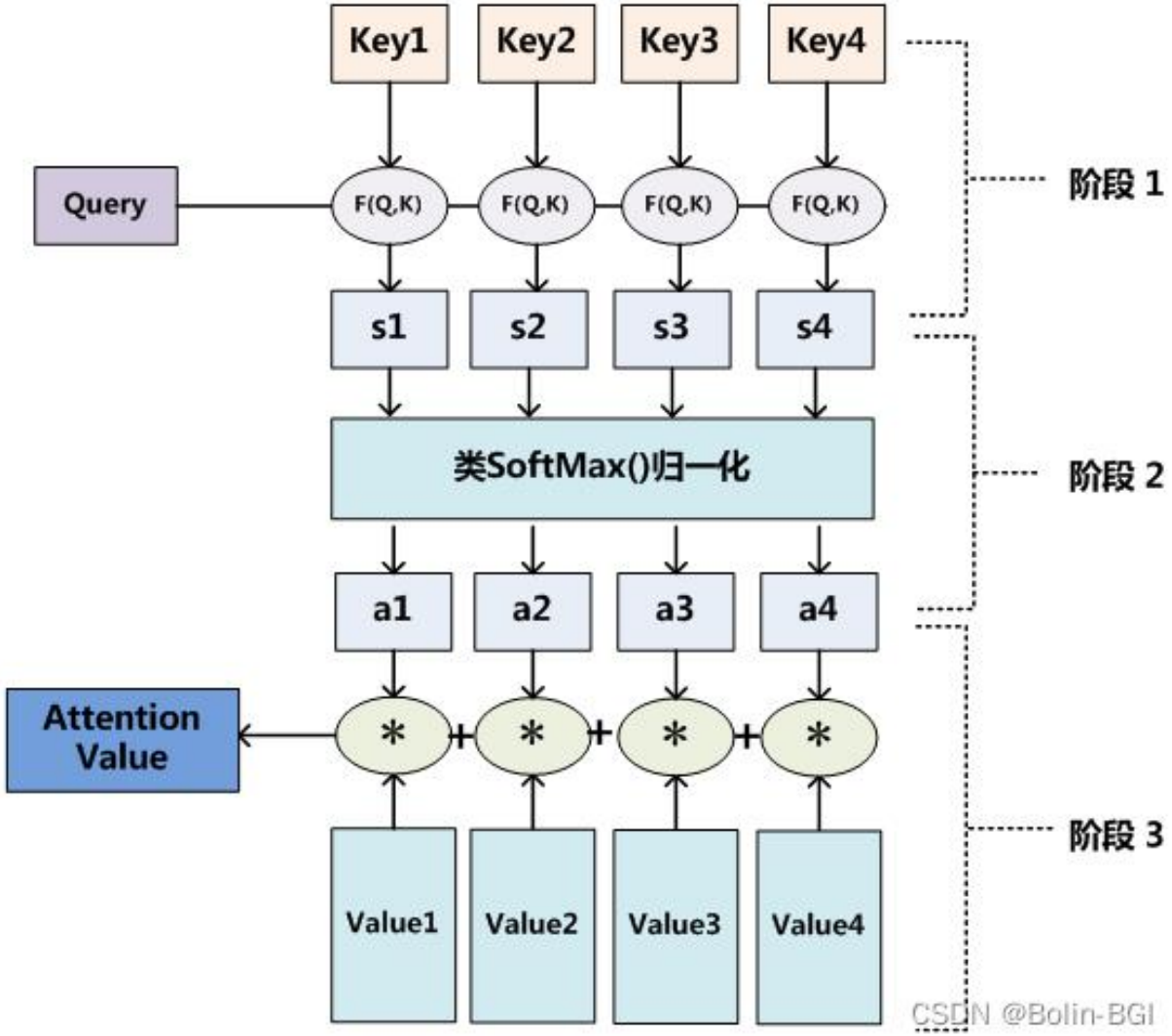
KIMI Linear:
A expressive, efficient attention architecture

研究人：龚平，任鑫

Context

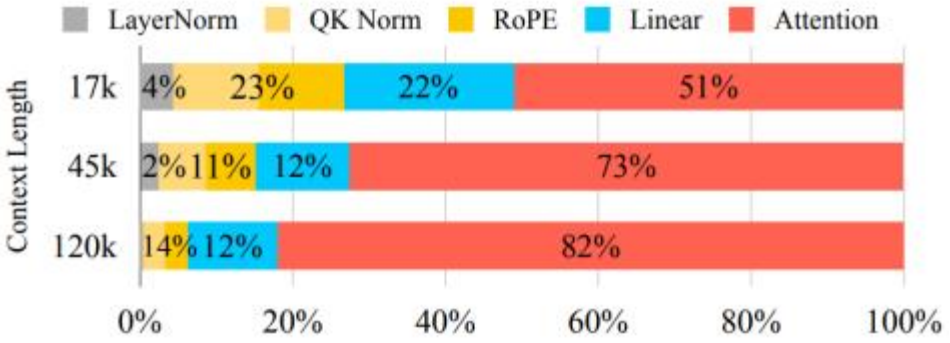
- ❑Problem of attention
- ❑What’s linear attention?
- ❑The history of linear attention
- ❑KIMI Linear
- ❑What’s next?

Problem of attention



时间复杂度：$O(N^2)$

空间负责度：$O(N)$



What's linear attention?

Attention

时间复杂度：$O(N^2)$

空间负责度：$O(N)$

Why not?

时间复杂度：$O(N)$

空间负责度：$O(1)$

The history of linear attention

❑Linear Attention演变

❖*Transformers are RNNs – 2020年*

$$\text{Attention}(Q,K,V)=\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

The history of linear attention

❑Linear Attention演变

❖*Transformers are RNNs – 2020年*

$$\text{Attention}(Q,K,V)=\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$\text{Attention}(Q,K,V)_i=\sum_{j=1}^{N}\frac{\exp(Q_i^T K_j)}{\sum_{k=1}^{N}\exp(Q_i^T K_k)}V_j$$

The history of linear attention

❑Linear Attention演变

❖*Transformers are RNNs – 2020年*

$$\text{Attention}(Q,K,V)=\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$\text{Attention}(Q,K,V)_i=\sum_{j=1}^{N}\frac{\exp(Q_i^T K_j)}{\sum_{k=1}^{N}\exp(Q_i^T K_k)}V_j$$

$$\text{Attention}(Q,K,V)_i=\frac{\sum_{j=1}^{N}\text{sim}(Q_i,K_j)V_j}{\sum_{j=1}^{N}\text{sim}(Q_i,K_j)}$$

Set: $\text{sim}(q,k)=\exp\left(\frac{q^T k}{\sqrt{D}}\right)$

The history of linear attention

❑Linear Attention演变

❖*Transformers are RNNs – 2020年*

$$V_i' = \text{Attention}(Q,K,V)_i = \frac{\sum_{j=1}^{N} \text{sim}(Q_i, K_j)V_j}{\sum_{j=1}^{N} \text{sim}(Q_i, K_j)}$$

Use $\phi()$ to replace sim()

$$V_i' = \frac{\sum_{j=1}^{N} \phi(Q_i)^T \phi(K_j)V_j}{\sum_{j=1}^{N} \phi(Q_i)^T \phi(K_j)} \Rightarrow V_i' = \frac{\phi(Q_i)^T \sum_{j=1}^{N} \phi(K_j)V_j^T}{\phi(Q_i)^T \sum_{j=1}^{N} \phi(K_j)}$$

The history of linear attention

❑Linear Attention演变

❖*Transformers are RNNs – 2020年*

$$V_i' = \frac{\phi(Q_i)^T \sum_{j=1}^{N} \phi(K_j) V_j^T}{\phi(Q_i)^T \sum_{j=1}^{N} \phi(K_j)} \Rightarrow \begin{cases} S_i = \sum_{j=1}^{i} \phi(K_j) V_j^T \\ Z_i = \sum_{j=1}^{i} \phi(K_j) \end{cases} \Rightarrow V_i' = \frac{\phi(Q_i)^T S_i}{\phi(Q_i)^T Z_i}$$

Update:

$$S_i = S_{i-1} + \phi(K_i) V_i^T$$

$$Z_i = Z_{i-1} + \phi(K_i)$$

Definition of RNNs: The output depends only on the current input and a fixed-size hidden state.

The history of linear attention

❑Linear Attention演变

❖*Transformers are RNNs – 2020年*

- ➢ *理论优雅*
- ➢ *缺乏 Softmax 的精准聚焦能力，导致语义混淆（权重分配不当）和过时信息难以擦除。*

$$V_i' = \frac{\phi(Q_i)^T S_i}{\phi(Q_i)^T Z_i}$$

$$S_i = S_{i-1} + \phi(K_i)V_i^T$$

$$Z_i = Z_{i-1} + \phi(K_i)$$

The history of linear attention

❑Linear Attention演变

- ❖*Transformers are RNNs – 2020年*
- ❖*RetNet – 2021年*
- ❖*DeltaNet – 2025年*
- ❖*Gated DeltaNet – 2025年*
- ❖*KIMI Linear – 2025年*

记忆更新模式（指数衰减）：

$$S_t = \gamma S_{t-1} + K^T V$$

- 原理：旧信息按比例衰减，新信息直接叠加。
- 优势：实现简单 + 容易并行化
- 问题：记忆容易受历史噪音的累积影响

The history of linear attention

❑Linear Attention演变

- ❖*Transformers are RNNs – 2020年*
- ❖*RetNet – 2021年*
- ❖*DeltaNet – 2024年*
- ❖*Gated DeltaNet – 2025年*
- ❖*KIMI Linear – 2025年*

记忆更新模式（指数衰减）：

$$S_t = S_{t-1} + \beta(V - S_{t-1}K)K^T$$

- 原理：计算预测误差，有选择地更新记忆
- 直观理解：先计算当前记忆对新输入的预测误差$(V - S_{t-1}K)$。
- 问题：理论上具有良好的效果。但是训练的时候，历史的梯度积累导致数值不稳定。有点像一个优化器

The history of linear attention

❑Linear Attention演变

- ❖*Transformers are RNNs – 2020年*
- ❖*RetNet – 2021年*
- ❖*DeltaNet – 2024年*
- ❖*Gated DeltaNet – 2025年*
- ❖*KIMI Linear – 2025年*

1. RetNet的长程衰减真的不好吗？
2. DeltaNet的修正机制真的好吗？

修正机制：

$$S_t = \gamma_t S_{t-1} + \beta(V - S_{t-1}K)K^T$$

The history of linear attention

❑Linear Attention演变

- ❖*Transformers are RNNs – 2020年*
- ❖*RetNet – 2021年*
- ❖*DeltaNet – 2024年*
- ❖*Gated DeltaNet – 2025年*
- ❖*KIMI Linear – 2025年*

修正机制：

$$S_t = \gamma_t S_{t-1} + \beta(V - S_{t-1}K)K^T$$

原理：该忘就忘 + 精确修改记忆

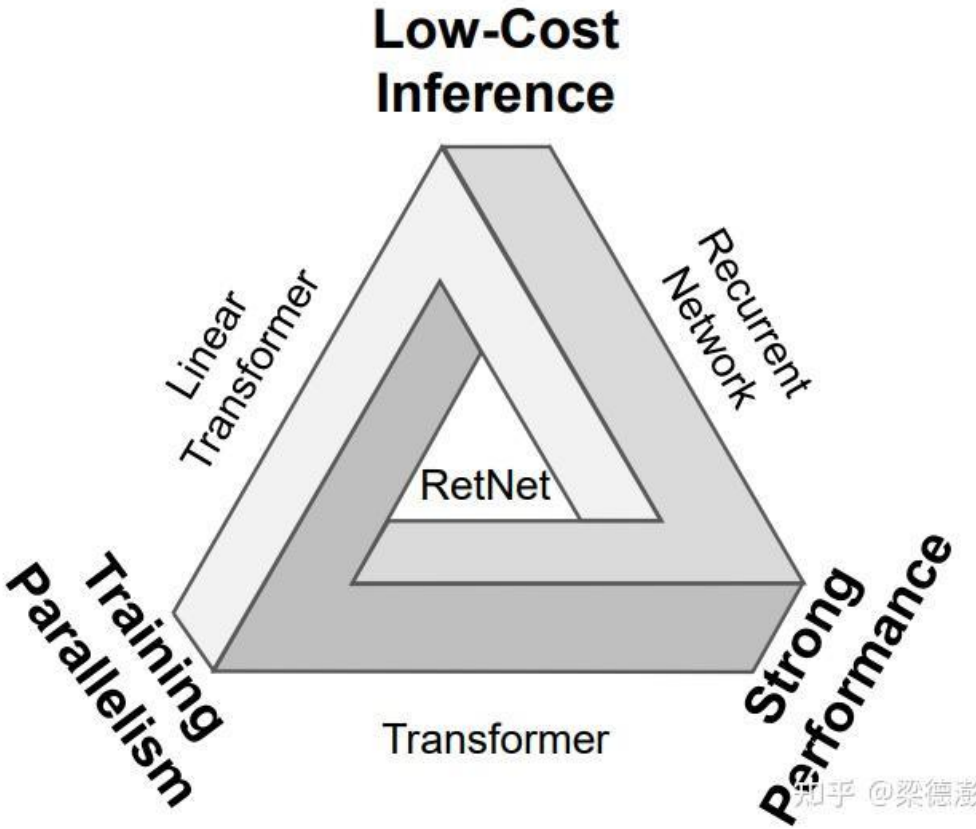
问题：代价是什么？

Parallelism Problem

❑三角问题

❖Gated DeltaNet

- $S_t = \gamma_t S_{t-1} + \beta(V - S_{t-1}K)K^T$
- S_t 依赖于 S_{t-1}，怎么训练并行化?



Parallelism Problem

❑线性化重写

$$S_t = \alpha_t S_{t-1} + \beta_t (v_t - S_{t-1} k_t) k_t^T$$

$$S_t = S_{t-1} \underbrace{(\alpha_t I - \beta_t k_t k_t^T)}_{A_t} + \underbrace{\beta_t v_t k_t^T}_{B_t}$$

$$S_t = S_{t-1} A_t + B_t$$

Parallelism Problem

❑结合率

$$S_t = S_{t-1}A_t + B_t$$

$$S_3 = ((S_0A_1 + B_1)A_2 + B_2)A_3 + B_3$$

关键矩阵： $A_1A_2A_3, B_1A_2A_3, B_2A_3, B_3$

关键技术：树状归约、前缀和算法

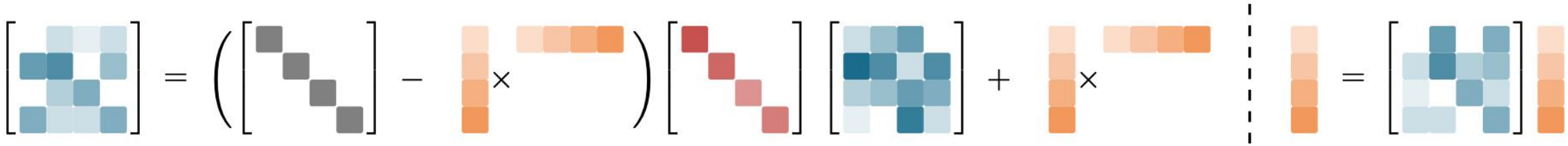
Others: System Problem

- 关键技术：树状归约、前缀和算法
- 大规模的Linear Attention训练真的简单吗?
  - 负载均衡问题?
  - 算子问题?
  - 通行和计算如何重叠?
- 长文的Linear Attention推理（Prefill）真的简单吗?

KIMI Linear - KDA

- KDA: 修正机制 (but channel-wise gate):
- $S_t = \gamma_t S_{t-1} + \beta(V - S_{t-1}K)K^T$

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

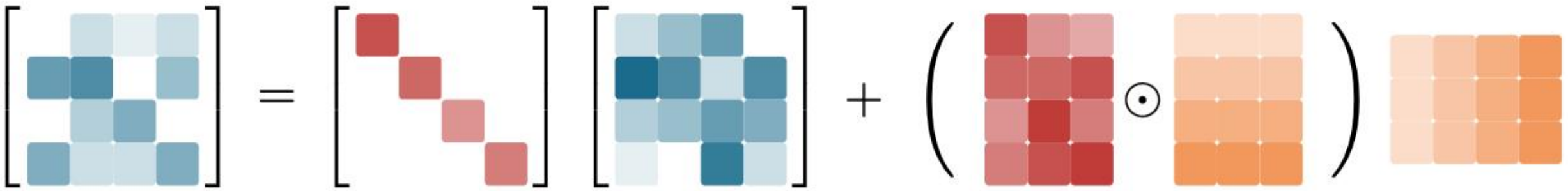


KIMI Linear - KDA

❑并行化问题

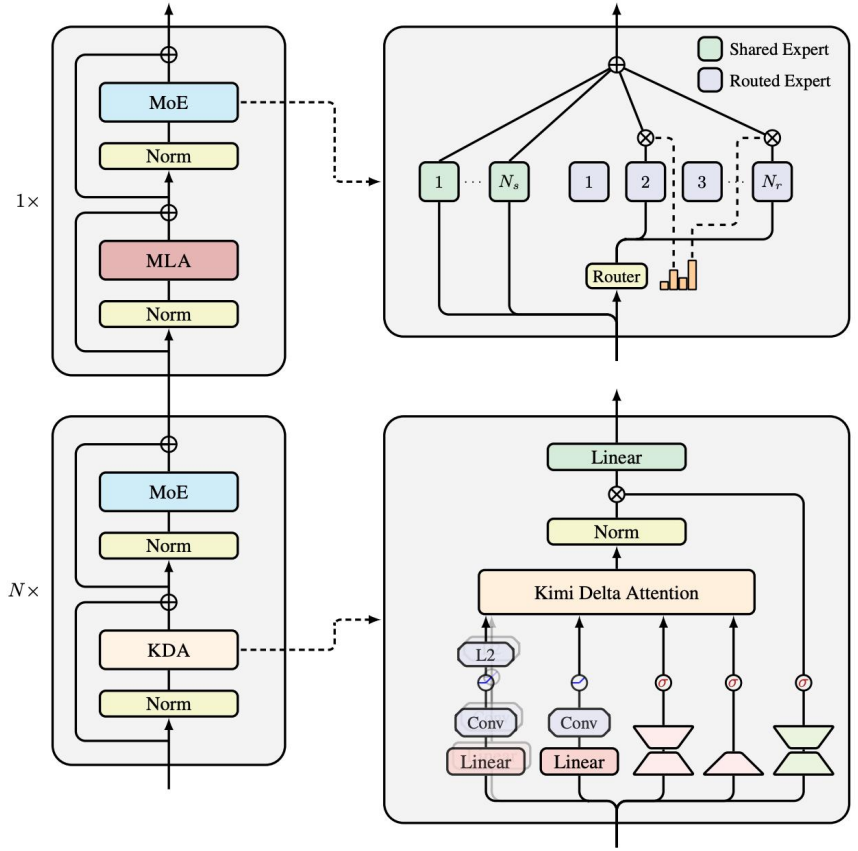
- ❖WY Representation
- ❖UT transform

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



KIMI Linear - Arch

- 3 KDA + 1 MLA



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

KIMI Linear - NoPE

- NoPE for MLA
- NoPE is not without PE
- KDA Layers as learnable position embeddings
  - Kimi Delta Attention (KDA) 模块本质上是一个递归模型（RNN-like）。它通过隐状态的迭代更新和门控衰减，将“位置信息”内化为了“状态信息”，从而让后续的全注意力层可以直接利用这些含时序特征的向量，无需外挂显式的位置编码。

KIMI Linear - NoPE

❑RoPE

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

$$\mathbf{R}_j^k = \begin{pmatrix} \cos(j\theta_k) & -\sin(j\theta_k) \\ \sin(j\theta_k) & \cos(j\theta_k) \end{pmatrix}$$

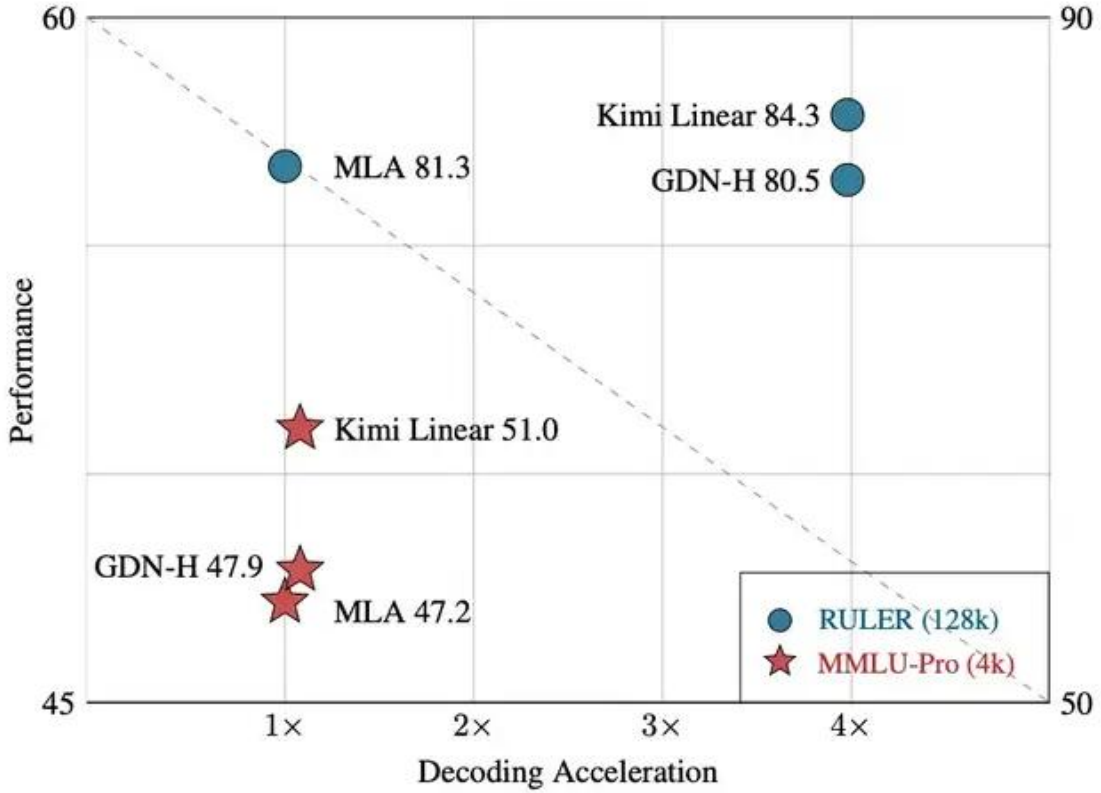
$$\text{s} \prod_{j=i+1}^{t} \mathbf{R}_j = \begin{pmatrix} \cos((t-i)\theta_k) & -\sin((t-i)\theta_k) \\ \sin((t-i)\theta_k) & \cos((t-i)\theta_k) \end{pmatrix}$$

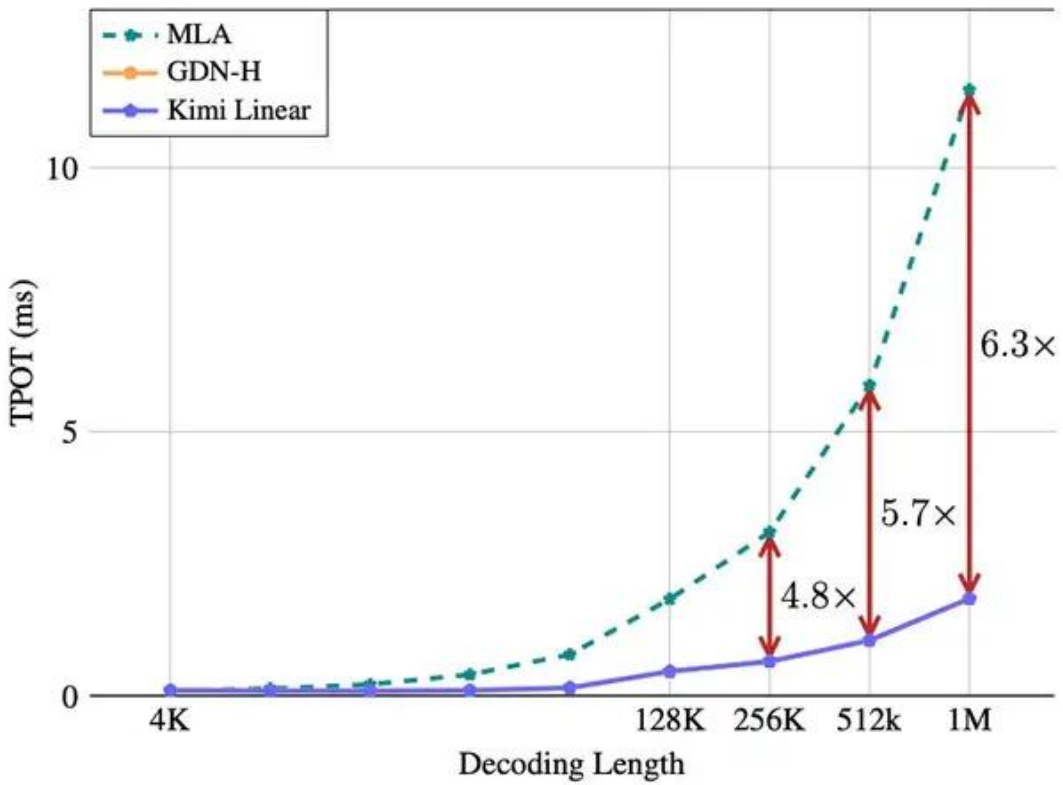
❑KDA

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

Results

❑精度和性能





Results

❑精度

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 1.4T 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**

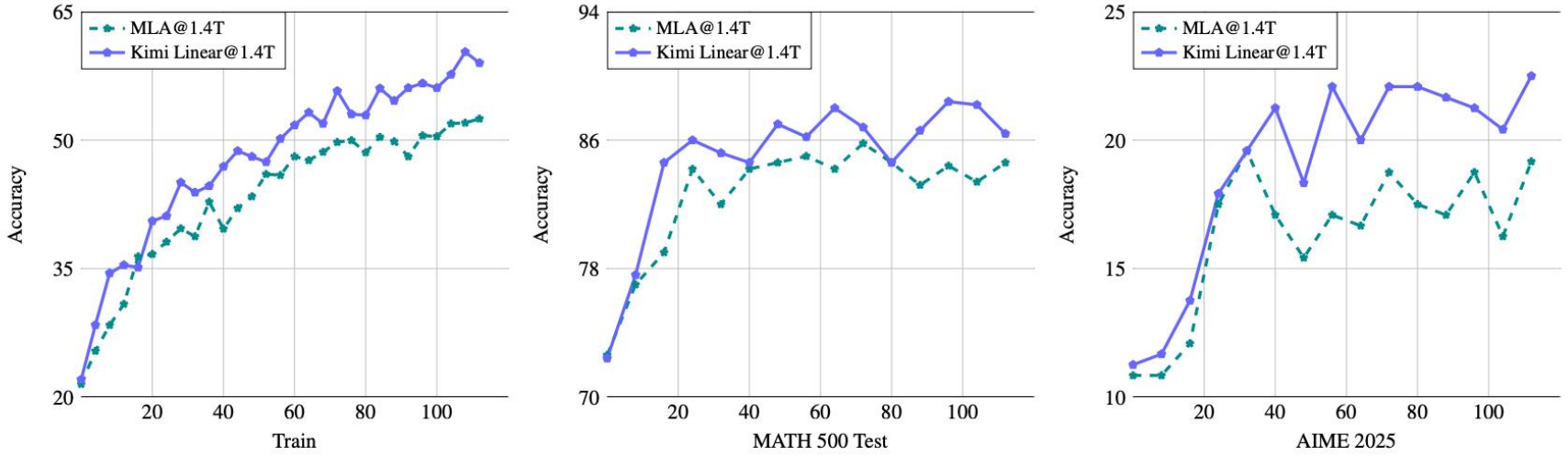


Figure 6: The training and test accuracy curves for Kimi Linear@1.4T and MLA@1.4T during Math RL training. Kimi Linear consistently outperforms the full attention baseline by a sizable margin during the whole RL process.

Results

❑消融实验

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 (↓)	Validation PPL (↓)
Hybrid ratio	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
w/o output gate		9.25	5.67
w/ swish output gate		9.43	5.81
w/o convolution layer		9.29	5.70

Conclusions

- KIMI Linear 是一种专为“超长上下文”设计的混合架构 LLM。
  - 架构设计： 采用 Hybrid（混合）策略，由大量的 Linear Attention（KDA） 层和MLA 层交替组成（通常比例为 3:1）。
  - 核心创新： 利用 GDN (Gated Delta Network) 的递归特性（类 RNN） 自动捕捉序列顺序，从而实现了 NoPE（无位置编码）。
  - 关键优势： 结合了线性注意力的高推理效率/低显存占用与标准注意力的高召回能力，完美解决了长文本任务中“性能”与“成本”的平衡难题。

What's Next?

❑About Linear Attention

2025.11.06

为什么MiniMax M2是一个Full Attention模型?

作为MiniMax M2预训练的负责人, 我收到了很多来自社区的询问："为什么你们在MiniMax M2上开倒车, 采用了Full Attention 机制？"在一次又一次的聊天中解释了背后的故事后, 我觉得是时候在一篇blog里写下我们的心路历程。

我可以花一整个下午来讨论为什么应该构建应该做 Linear/Sparse Attention。同样, 我也可以反过来花一整个下午来讨论为什么不应该去做。但所有这些纸上谈兵又有什么意义呢? 回到实际情况里, 我们要不要做呢?

先说结论: 我们一直在研究它，但在一个现实的工业系统中, Efficient Attention想要打败Full Attention还有些

What's Next?

❑About Linear Attention

❖观测成本高

- ➢不scale上去, 永远不知道会发生什么, 很多问题在小规模试验中无法暴露。
- ➢发现问题, 真的比解决问题要难得的多。

❖相比Full Attention, Linear Attention和Sparse Attention的基建要差的多, 想要真的拿到收益, 要补不少课。

- ➢States的低精度存储: 当前Linear Attention对精度要求比Full Attention高得多;
- ➢如何解决Prefix Cache: 正常业务命中Cache的概念是很高的;
- ➢如何优化Linear Attention上的投机解码
- ➢Linear Attention的训练优化问题 - 算子/调度