Exploring Efficient Architectures for LLMs

Department: Theory Lab, Huawei Hong Kong Research Center

Author: Wenqi ZENG

Date: 18/11

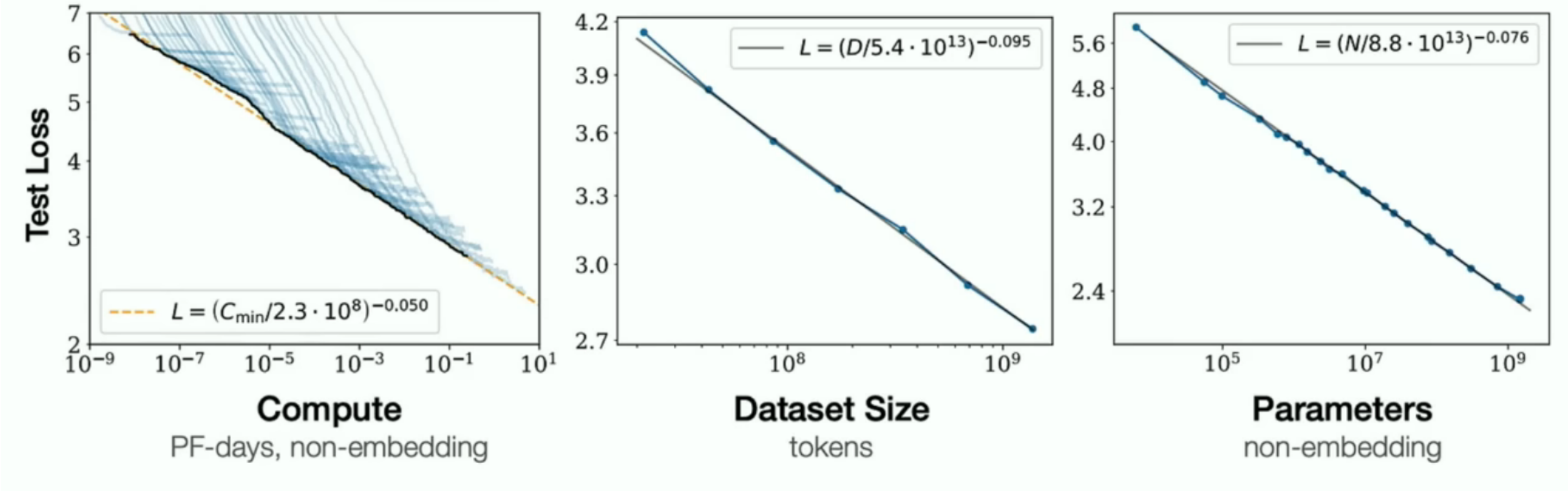
Security Level:

HUAWEI

The age of Pre-Training LLMs

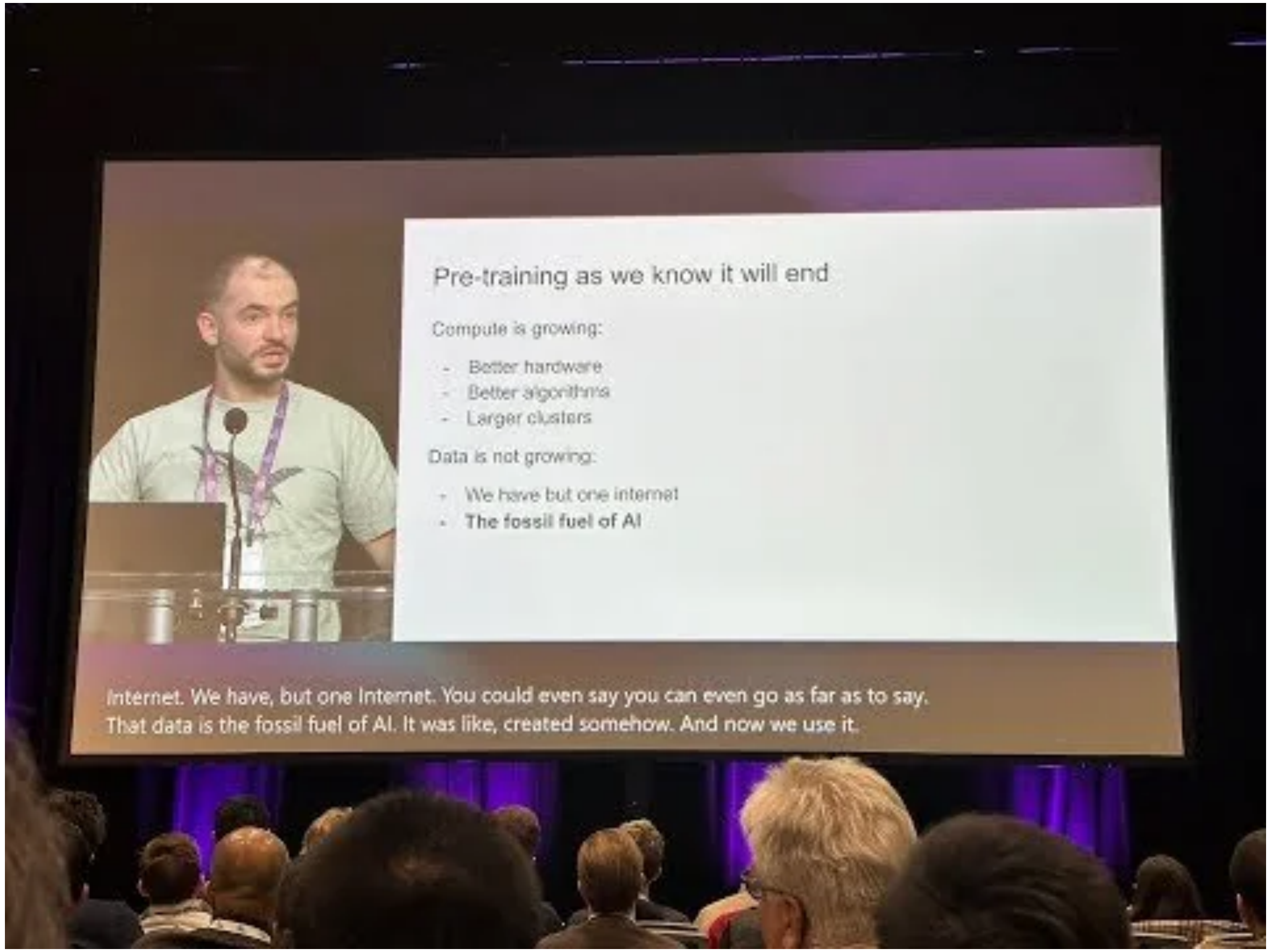
Autoregressive models trained on text, big network, big datasets

- GPT-2 [Radford et al., 2019]
- GPT-3 [Brown et al., 2020]
- Scaling laws [Kaplan et al., 2020]



https://www.youtube.com/watch?v=1yvBqasHLZs

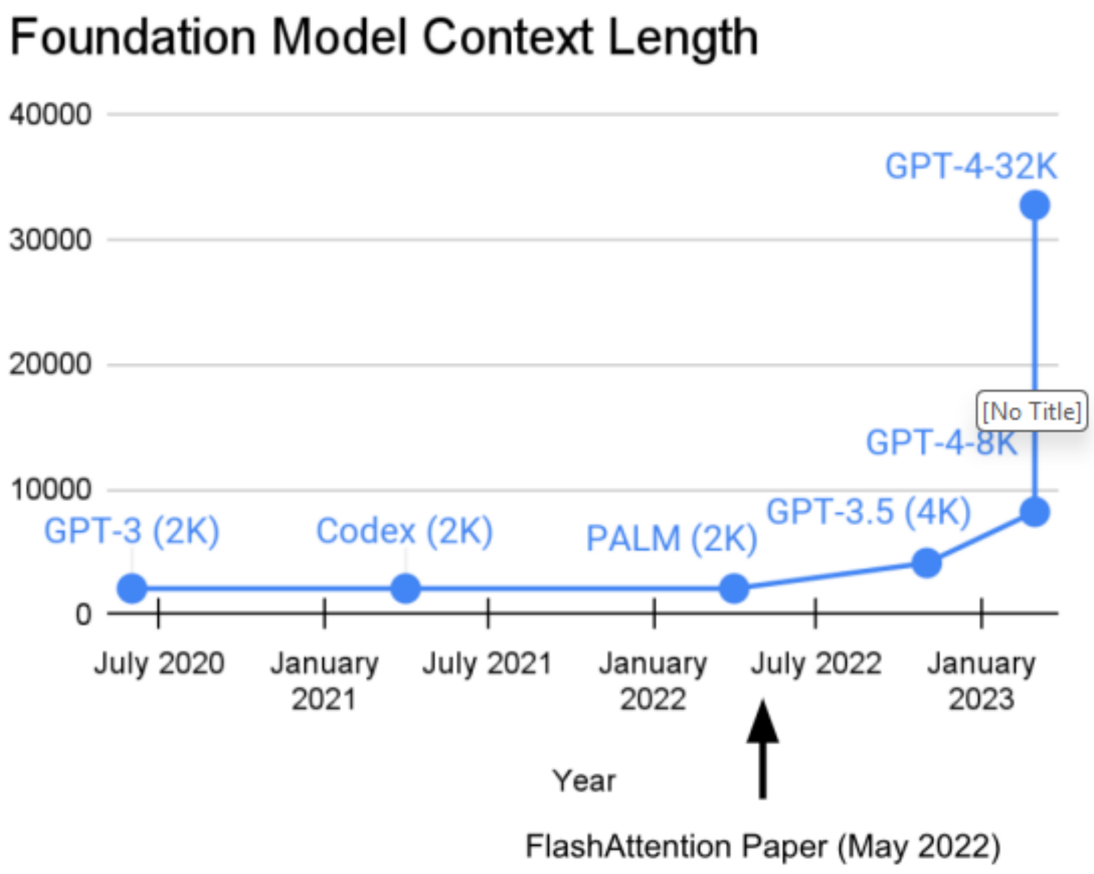
Why we need new Architecture for LLM



https://www.youtube.com/watch?v=1yvBqasHLZs

- **Training Scaling Hits a Wall**
  - Marginal returns from data and compute are diminishing.
- **Test-Time Scaling Emerges**
  - Amplifies the Transformer's core efficiency problems.
- **The Need for New Architectures**
  - Must solve both training and test-time challenges simultaneously

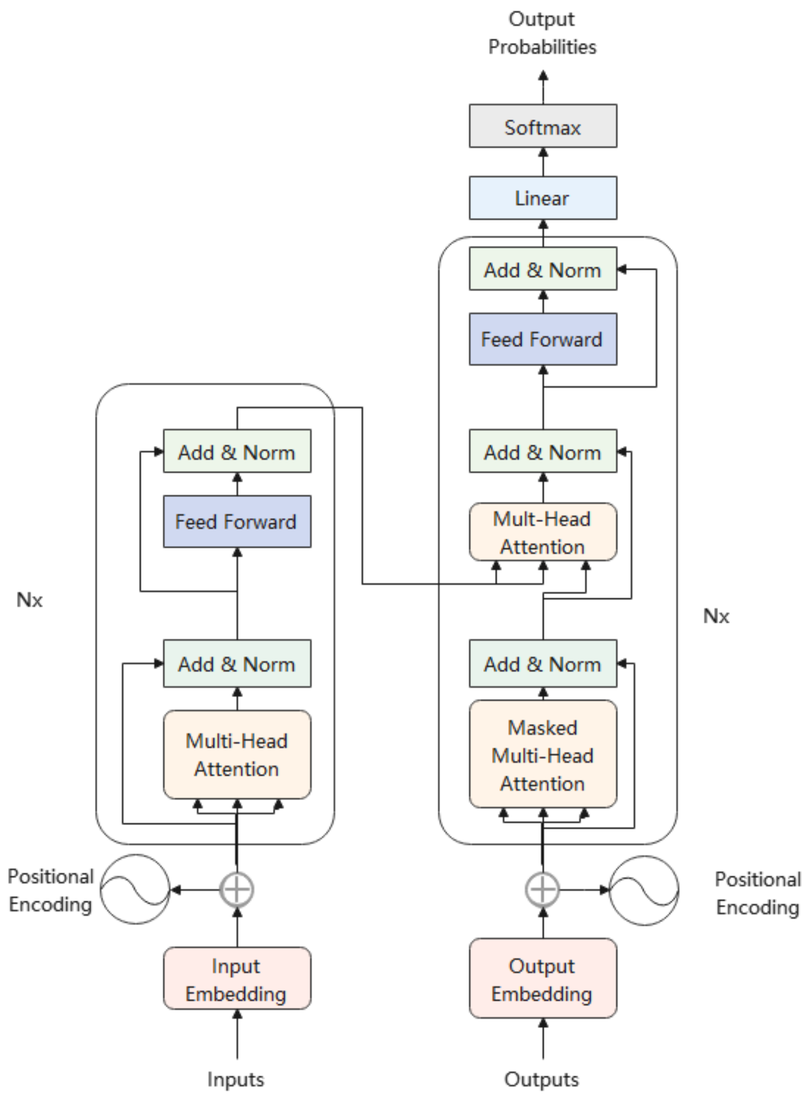
Why we need new Architecture for LLM



- **Training Scaling Hits a Wall**
  - Marginal returns from data and compute are diminishing.
- **Test-Time Scaling Emerges**
  - Amplifies the Transformer's core efficiency problems.
- **The Need for New Architectures**
  - Must solve both training and test-time challenges simultaneously

https://www.youtube.com/watch?v=1yvBqasHLZs

Behind LLM: Transformer Architecture



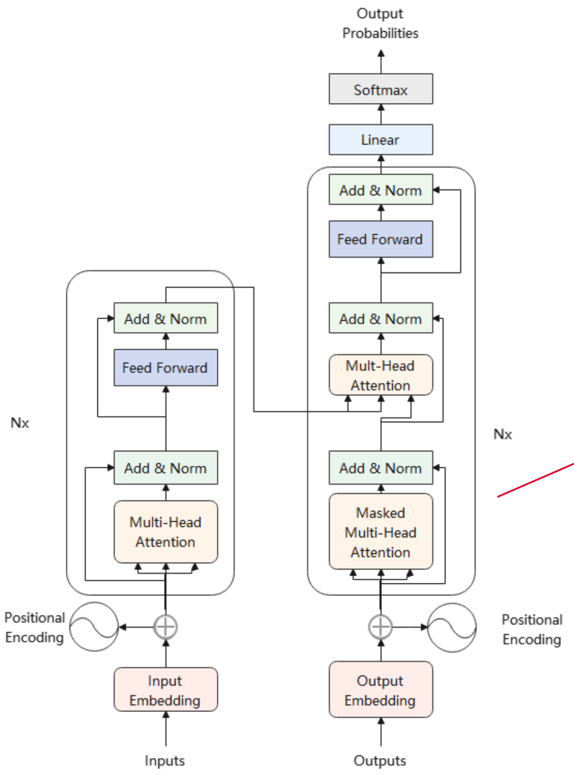
- Training: quadratic time complexity $O(L^2)$
  - Expensive for long sequence modeling (e.g., video or DNA modeling)s

- Inference: Linear memory complexity $O(L)$
  - requires storing KV cache for each token
  - High memory burdens

Behind LLM: Transformer Architecture

time complexity mainly comes from dot-product softmax attention

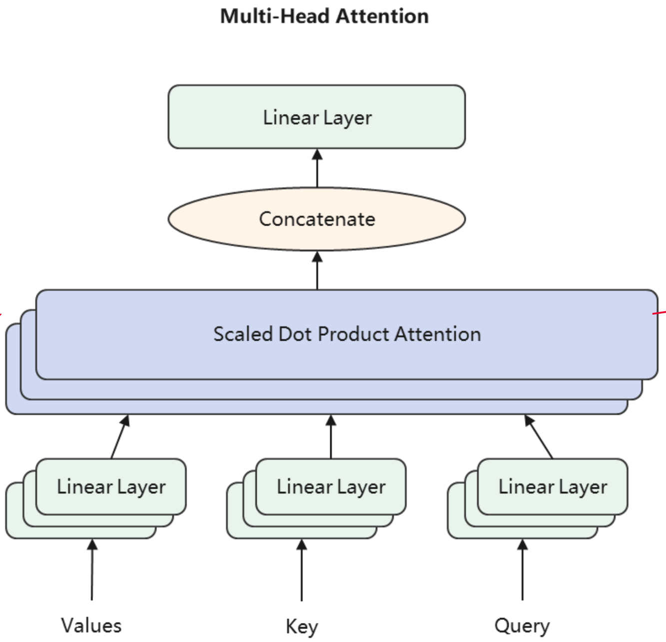
Transformer Architecture



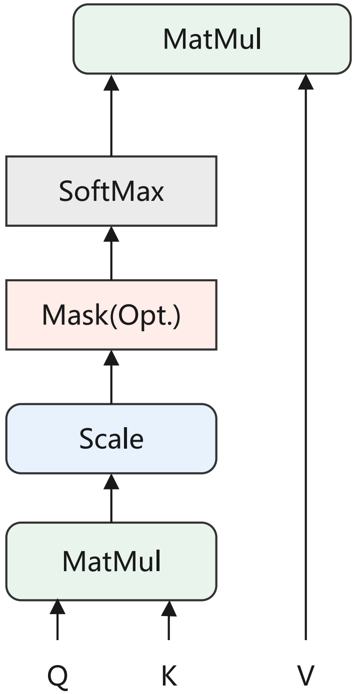
Transformer: $\mathbb{R}^L \rightarrow \mathbb{R}^L$

- L: Sequence length

Multi-Head Attention Architecture



Dot-Product Attention Architecture



$$Q = xW_Q \in \mathbb{R}^{L\times d}$$
$$K = xW_K \in \mathbb{R}^{L\times d}$$
$$V = xW_V \in \mathbb{R}^{L\times d}$$

$$O \coloneqq \mathrm{softmax}(QK^T \odot M)V \ \in \mathbb{R}^{L\times d} \sim O(L^2)$$

Outline – Efficient Attention Variants

1. Linear Attention Machenism

- **Data-dependent decay:** RetNet, LighteningAttention, Mamba2, GLA
- **Test time online learning:** DeltaNet, Test-Time-Training, Titans, RWKV7, Gated DeltaNet

2. Sparse Attention Mechanisms

- **Static Sparsity:** BigBird, StreamingLLM, H2O
- **Dynamic Sparsity**: Native Sparse Attention (DeepSeek), MoBA (Kimi)

3. Hybrid Attention Mechanisms

- **Inter-layer mixing**: Minimax-01, Jamba, Samba
- **Intra-layer mixing:** Hymba

1.1 Linear Attention : From standard attention

Linear Attention = Standard attention - softmax

Softmax attention:

$$\text{Parallel training}: \quad \mathbf{O} = \text{softmax}(\mathbf{Q}\mathbf{K}^\top \odot \mathbf{M})\mathbf{V} \quad \in \mathbb{R}^{L\times d}$$

$$\text{Iterative inference}: \quad \mathbf{o_t} = \sum_{j=1}^{t} \frac{\exp(\mathbf{q}_t^\top \mathbf{k}_j)}{\sum_{l=1}^{t} \exp(\mathbf{q}_t^\top \mathbf{k}_l)} \mathbf{v}_j \quad \in \mathbb{R}^{d}$$

where $\mathbf{M} \in \mathbb{R}^{L\times L}$ is the casual mask:

$$\mathbf{M}_{i,j} = \begin{cases} -\infty & \text{if } j > i \\ 1 & \text{if } j \le i \end{cases}$$

1.1 Linear Attention : From standard attention

Linear Attention = Standard attention - softmaxs

Linear attention (Katharopoulos et al. 2020):

$$\text{Parallel training}: \quad \mathbf{O} = \cancel{\mathrm{softmax}}(\mathbf{Q}\mathbf{K}^\top \odot \mathbf{M})\mathbf{V} \qquad \in \mathbb{R}^{L \times d}$$

$$\text{Iterative inference}: \quad \mathbf{o_t} = \sum_{j=1}^{t} \frac{\cancel{\exp}(\mathbf{q}_t^\top \mathbf{k}_j)}{\cancel{\sum_{l=1}^{t} \exp(\mathbf{q}_t^\top \mathbf{k}_l)}} \mathbf{v}_j \qquad \in \mathbb{R}^d$$

where the denominator is harmful for linear attention's training stability and performance (Qin et al. 2022). Therefore, nearly all recent linear attention models remove this normalization term.

1.1 Linear Attention : From standard attention

Linear Attention = Standard attention - softmax

Linear attention (Katharopoulos et al. 2020):

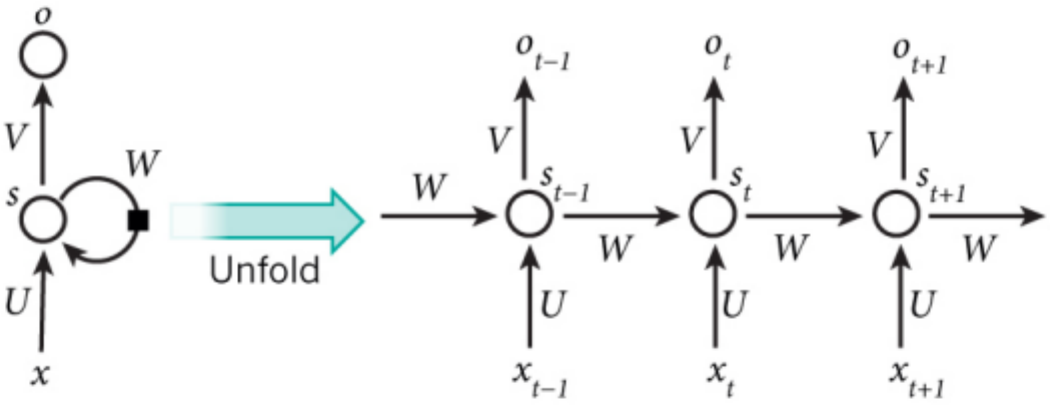
$$\text{Parallel training :} \quad \mathbf{O} = \cancel{\text{softmax}}(\mathbf{Q}\mathbf{K}^\top \odot \mathbf{M})\mathbf{V} \quad \in \mathbb{R}^{L \times d}$$

$$\text{Iterative inference :} \quad \mathbf{o_t} = \sum_{j=1}^{t} \frac{\cancel{\exp}(\mathbf{q}_t^\top \mathbf{k}_j)}{\cancel{\sum_{l=1}^{t} \exp(\mathbf{q}_t^\top \mathbf{k}_l)}} \mathbf{v}_j \quad \in \mathbb{R}^{d}$$

We abuse the notation $\mathbf{M}$ to denote the causal mask for both softmax and linear attention. Here we have:

$$\mathbf{M}_{i,j} = \begin{cases} 0 & \text{if } j > i \\ 1 & \text{if } j \leq i \end{cases}$$

1.2 Linear Attention: From RNN view



Revisit RNN:

- Training: linear complexity $O(L)$, however, traditional RNNs are not parallelizable.
- Inference: constant memory $O(1)$

1.2 Linear Attention: From RNN view

Linear Attention = Linear RNN + matrix-valued hidden states

$$
\begin{aligned}
\mathbf{o_t} &= \sum_{j=1}^{t} (\mathbf{q}_t^\top \mathbf{k}_j)\mathbf{v}_j \\
&= \sum_{j=1}^{t} \mathbf{v}_j(\mathbf{k}_j^\top \mathbf{q}_t) \quad \mathbf{k}_j^\top \mathbf{q}_t = \mathbf{q}_t^\top \mathbf{k}_j \in \mathbb{R} \\
&= \underbrace{\Big(\sum_{j=1}^{t} \mathbf{v}_j \mathbf{k}_j^\top\Big)}_{\mathbf{S}_t \in \mathbb{R}^{d \times d}} \mathbf{q}_t \qquad \text{By associativity}
\end{aligned}
$$

Let $\mathbf{S}_t = \sum_{j=1}^{t} \mathbf{v}_j \mathbf{k}_j^\top \in \mathbb{R}^{d \times d}$ be the matrix-valued hidden state, then:

$$
\begin{aligned}
\mathbf{S}_t &= \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top \qquad && \in \mathbb{R}^{d \times d} \\
\mathbf{o}_t &= \mathbf{S}_t \mathbf{q}_t && \in \mathbb{R}^{d}
\end{aligned}
$$

- Linear attention implements **elementwise linear recurrence**.
- Linear attention has a **matrix-valued hidden state**, significantly increasing the state size.

1.3 Challenges in Linear Attention: Instability

$$\mathbf{S}_t = \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top \qquad \in \mathbb{R}^{d \times d}$$
$$\mathbf{o}_t = \mathbf{S}_t \mathbf{q}_t \qquad \in \mathbb{R}^{d}$$

- **Instability:** the hidden state value could explode due to cumulative sum without decay
- **Poor performance:** vanilla linear attention models significantly underperform Transformers in language modeling perplexity

1.3 Challenges in Linear Attention: Instability

A simple fix: linear attention with constant decay

$$\mathbf{S}_t = \gamma \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top \qquad \in \mathbb{R}^{d \times d}$$
$$\mathbf{o}_t = \mathbf{S}_t \mathbf{q}_t \qquad \in \mathbb{R}^{d}$$

- γ is a constant exponential decay factor $0 < \gamma < 1$.
- Works well in practice: RetNet (Sun et al. 2023), Lightning Attention (Qin et al. 2024b)
- Lacking selectivity: a potential issue.

1.3 Challenges in Linear Attention: Instability

A simple fix: linear attention with data-dependent decay

$$\mathbf{S}_t = \gamma_t \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top \qquad \in \mathbb{R}^{d \times d}$$

$$\mathbf{o}_t = \mathbf{S}_t \mathbf{q}_t \qquad \in \mathbb{R}^{d}$$

- $\gamma_t \in (0, 1)$ is a data-dependent decay term
- Enables dynamic control of memory retention/forgetting based on input data.
- Examples: Mamba2 (Dao and Gu 2024), mLSTM (Beck et al. 2024), Gated Retention (Sun et al. 2024b).

1.3 Challenges in Linear Attention: Poor Performance

A complicated fix: Linear attention optimizes a negative linear inner product loss via SGD

The objective predicts the target value $\mathbf{v}_t$ by transforming the key $\mathbf{k}_t$ with $\mathbf{S}$.

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

Performing a single step of SGD:

$$\begin{aligned} \mathbf{S}_t &= \mathbf{S}_{t-1} - \beta_t \nabla \mathcal{L}_t(\mathbf{S}_{t-1}) \\ &= \mathbf{S}_{t-1} + \beta_t \mathbf{v}_t \mathbf{k}_t^\top \end{aligned}$$

- Learning rate $\beta_t = 1$ recovers vanilla linear attention.
- Mamba2's update rule $\mathbf{S}_t = \alpha_t \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top$ can be interpreted as online SGD with weight decay α_t.

1.3 Challenges in Linear Attention: Poor Performance

A complicated fix: Linear attention optimizes a online regression loss via SGD

Online regression loss is better for predicting $\mathbf{v}_t$ from $\mathbf{k}_t$ and $\mathbf{S}_{t-1}$.

$$\mathcal{L}_t(\mathbf{S}) = \frac{1}{2}\|\mathbf{S}\mathbf{k}_t - \mathbf{v}_t\|^2$$

Performing a single step of SGD:

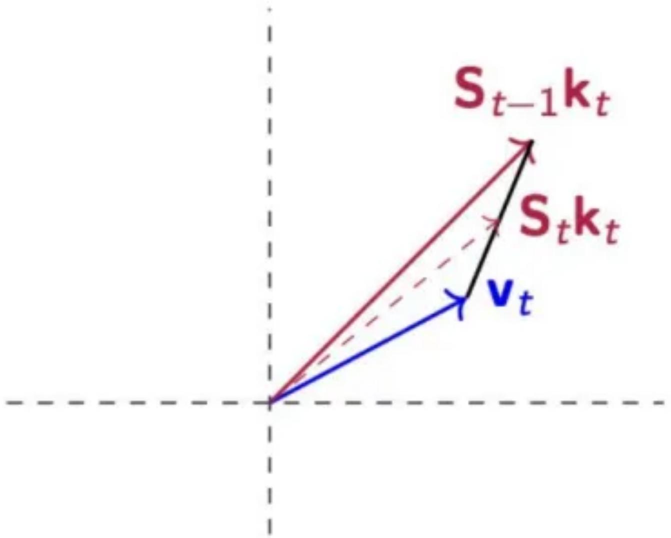
$$\begin{aligned} \mathbf{S}_t &= \mathbf{S}_{t-1} - \beta_t \nabla \mathcal{L}_t(\mathbf{S}_{t-1}) \\ &= \mathbf{S}_{t-1} - \beta_t \left(\mathbf{S}_{t-1}\mathbf{k}_t - \mathbf{v}_t\right) \mathbf{k}_t^\top \end{aligned}$$

- When $\beta_t \in (0, 1)$, the DeltaNet update rule (Schlag, Irie, and Schmidhuber 2021; Yang et al. 2024) is recovered.

1.3 Challenges in Linear Attention: Poor Performance

A complicated fix: Linear attention optimizes a online regression loss via SGD

Directly minimize Euclidean distance



Objective: $\mathcal{L}_t(\mathbf{S}) = \frac{1}{2}\|\mathbf{S}\mathbf{k}_t - \mathbf{v}_t\|^2$

SGD update: $\mathbf{S}_t = \mathbf{S}_{t-1} - \beta_t \nabla \mathcal{L}_t(\mathbf{S}_{t-1}) = \mathbf{S}_{t-1} - \beta_t(\mathbf{S}_{t-1}\mathbf{k}_t - \mathbf{v}_t)\mathbf{k}_t^\top$

1.3 Challenges in Linear Attention: Engineering Optimization

Parallel Form:

$$\mathbf{O} = (\mathbf{Q}\mathbf{K}^\top \odot \mathbf{M})\mathbf{V} \in \mathbb{R}^{L\times d}$$

- Still quadratic in sequence length

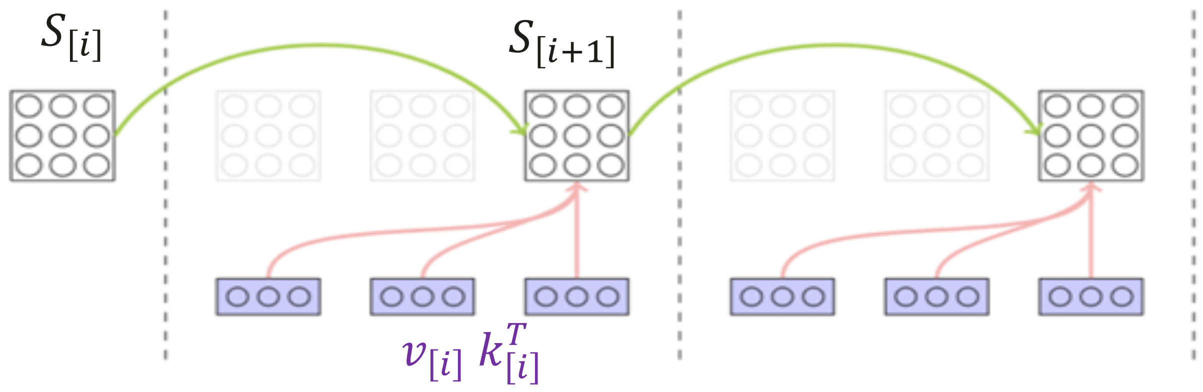
Recurrent Form:

$$\begin{aligned} \mathbf{S}_t &= \mathbf{S}_{t-1} + \mathbf{v}_t\mathbf{k}_t^\top && \in \mathbb{R}^{d\times d} \\ \mathbf{o}_t &= \mathbf{S}_t\mathbf{q}_t && \in \mathbb{R}^{d} \end{aligned}$$

- Sequential computation limits parallelization opportunities.
- Poor GPU utilization due to lack of matrix-multiply operations (even with parallel scan algorithms)

1.3 Challenges in Linear Attention: Engineering Optimization

Engineering Optimization of Linear Architecture: Chunk-wise parallelism



Input $X \in \mathbb{R}^{L \times F}$ is divided into C chunks, $S_{[i]} \in \mathbb{R}^{d \times d}$ represent the i-th chunk, $v_{[i]} \in \mathbb{R}^{C \times d}$, $i \in \{0, 1, \dots, \frac{L}{C}\}$

$$S_{[i+1]} = S_{[i]} + k_{[i]}^T v_{[i]}$$

- Computation within chunks can be parallelized, and the computational complexity of updating $S_{[i]}$ is: $O\left(\frac{L}{C} * C^2 d\right) = O(LCd)$
- Becomes standard for training modern linear attention models (e.g., Mamba2, Based, GLA, DeltaNet, Lightning Attention, mLSTM $\cdots$)

1.4 Summary

Tasks	SoftMax Attention	Linear Attention
Parallelized Training	$O = softmax\ (QK^T)V \ \in \mathbb{R}^{L\times d}$	$O = (QK^T)V$
Iterative Inference	$o_t = \sum_{j=1}^{t} \frac{\exp(q_t^T k_j)}{\sum_{l=1}^{t} \exp(q_l^T k_j)} v_j \in \mathbb{R}^{\mathrm{d}}$	$o_t = \sum_{j=1}^{t} (q_t^T k_j) v_j = \sum_{j=1}^{t} v_j\ (k_j^T q_t) = (\sum_{j=1}^{t} v_j\ k_j^T) q_t$
Storage	$\{k_i, v_i\}_{i=1\cdots t}$	$S_t = S_{t-1} + v_t\ k_t^T \in \mathbb{R}^{d\times d}$
Time Complexity	Single-step Computational Complexity O(L)	Single-step complexity O(1)
Space complexity	Single-step O(Ld)	Single-step O(d^2)

Note:

- S_tWithout a forgetting mechanism, numerical values can easily explode [2]
- S_t lacks In-context retrieval capability [2]

[1] : https://arxiv.org/abs/2210.10340
[2]: https://arxiv.org/abs/2406.06484s

1.4 Comparison of Algorithms for Linear Attention

Method	Memory Update Rule
Linear Attn	$\boldsymbol{M}_t = \boldsymbol{M}_{t-1} + \boldsymbol{k}_t^T \boldsymbol{v}_t$
Lightning	$\boldsymbol{M}_t = \gamma \boldsymbol{M}_{t-1} + \boldsymbol{k}_t^T \boldsymbol{v}_t$
RetNet	$\boldsymbol{M}_t = \gamma \boldsymbol{M}_{t-1} + \boldsymbol{k}_t^T \boldsymbol{v}_t$
GLA	$\boldsymbol{M}_t = (\boldsymbol{a}_t^T \mathbf{1}) \boldsymbol{M}_{t-1} + \boldsymbol{k}_t^T \boldsymbol{v}_t$
DeltaNet	$\boldsymbol{M}_t = (\boldsymbol{I} - \boldsymbol{k}_t^T \boldsymbol{k}_t) \boldsymbol{M}_{t-1} + b_t \boldsymbol{k}_t^T \boldsymbol{v}_t$
G-DeltaNet	$\boldsymbol{M}_t = a_t (\boldsymbol{I} - \boldsymbol{k}_t^T \boldsymbol{k}_t) \boldsymbol{M}_{t-1} + b_t \boldsymbol{k}_t^T \boldsymbol{v}_t$
TTT	$\boldsymbol{M}_t = \boldsymbol{M}_{t-1} + b_t \nabla l(\boldsymbol{M}_{t-1}; \boldsymbol{k}_t, \boldsymbol{v}_t)$
Titans*	$\boldsymbol{M}_t = a_t \boldsymbol{M}_{t-1} + b_t \nabla l(\boldsymbol{M}_{t-1}; \boldsymbol{k}_t, \boldsymbol{v}_t)$
Mamba2	$\boldsymbol{M}_t = a_t \boldsymbol{M}_{t-1} + b_t \boldsymbol{k}_t^T \boldsymbol{v}_t$
HGRN2	$\boldsymbol{M}_t = (\boldsymbol{a}_t^T \mathbf{1}) \boldsymbol{M}_{t-1} + (1 - \boldsymbol{a}_t)^T \boldsymbol{v}_t$
RWKV6	$\boldsymbol{M}_t = a_t \boldsymbol{M}_{t-1} + \boldsymbol{k}_t^T \boldsymbol{v}_t$
RWKV7	$\boldsymbol{M}_t = a_t \boldsymbol{M}_{t-1} + b_t \nabla l(\boldsymbol{M}_{t-1}; \boldsymbol{k}_t, \boldsymbol{v}_t)$

MoM Table 1: Comparison of Different Linear Models. M is same as S in previous slides.
https://arxiv.org/abs/2502.13685

Core Improvement: Introducing exponential decay
Drawback: γ is data-agnostic and lacks selectivity toward the data (improved by GLA, Mamba2).

Core Improvements: Modeling the update rule of M from the perspective of SGD; enhancing the model's in-context retrieval capability;
Delta Net: S updating equals SGD optimization for s $L_t(M) = \frac{1}{2}||Sk_t - v_t||^2$
G-Delta Net: Incorporating α_t, which is equivalent to adding an SGD momentum term

Core Improvement: Adding nonlinear terms to the loss function $L_t(S)$, $L_t(S) = \frac{1}{2}||f(S, k_t) - v_t||^2$ enhances the model's expressive power through nonlinear modeling.
Drawback: Nonlinear iterative form makes training difficult to parallelize.

1.5 Experimental Result

i. Language task performance: Linear models can match the performance of Transformers on some tasks.

Model	Wiki. ppl ↓	LMB. ppl ↓	LMB. acc ↑	PIQA acc ↑	Hella. acc_n ↑	Wino. acc ↑	ARC-e acc ↑	ARC-c acc_n ↑	SIQA acc ↑	BoolQ acc ↑	Avg.
Recurrent models											
RetNet	19.08	17.27	40.52	70.07	49.16	54.14	67.34	33.78	**40.78**	60.39	52.02
HGRN2	19.10	17.69	39.54	70.45	49.53	52.80	69.40	35.32	40.63	56.66	51.79
Mamba	17.92	15.06	43.98	71.32	52.91	52.95	69.52	35.40	37.76	**61.13**	53.12
Mamba2	16.56	12.56	45.66	71.87	55.67	55.24	**72.47**	37.88	40.20	60.13	54.89
DeltaNet	17.71	16.88	42.46	70.72	50.93	53.35	68.47	35.66	40.22	55.29	52.14
Gated DeltaNet	**16.42**	**12.17**	**46.65**	**72.25**	**55.76**	**57.45**	71.21	**38.39**	40.63	60.24	**55.32**
Attention or hybrid models											
Transformer++	18.53	18.32	42.60	70.02	50.23	53.51	68.83	35.10	40.66	57.09	52.25
Samba	16.13	13.29	44.94	70.94	53.42	55.56	68.81	36.17	39.96	62.11	54.00
Gated DeltaNet-H1	16.07	**12.12**	47.73	**72.57**	56.53	**58.40**	71.75	**40.10**	41.40	**63.21**	**56.40**
Gated DeltaNet-H2	**15.91**	12.55	**48.76**	72.19	**56.88**	57.77	71.33	39.07	**41.91**	61.55	56.18

Table 3: Performance comparison on language modeling and zero-shot common-sense reasoning.

ii. Training speed (+chunk-wise parallel training). As the sequence length increases, there is a 1–2x performance gain compared to Transformer.

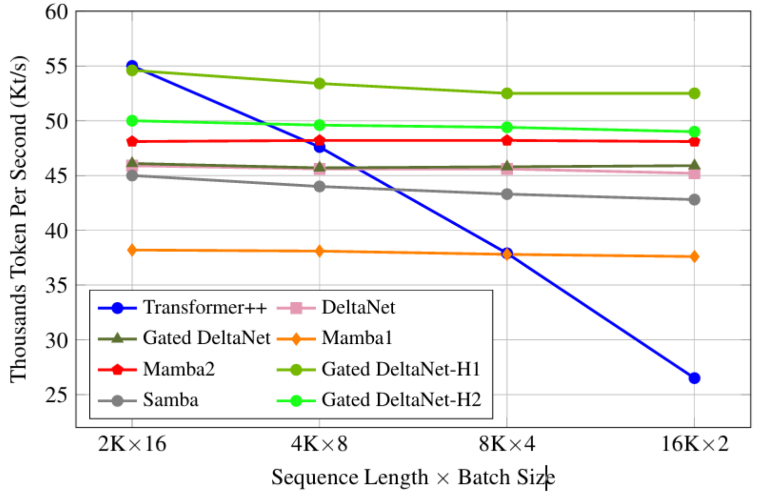


Figure 3: Training throughput comparison of 1.3B models on a single H100 GPU.

Source: Gated DeltaNet, https://arxiv.org/pdf/2412.06464

1.5 Experimental Result

iii. Stability of training on long sequences (+ decay): Token loss corresponding to sequence positions; linear models are not affected by position.

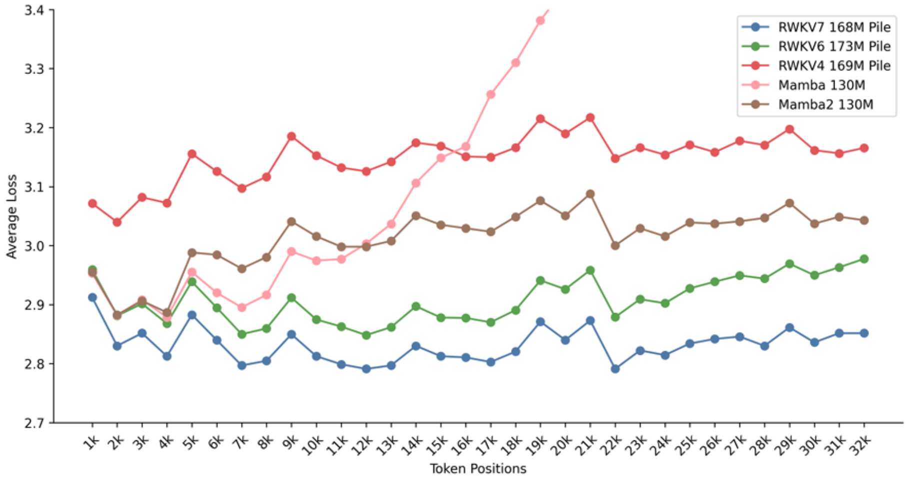


Figure 5: PG19 loss versus sequence position for RWKV and Mamba models trained on The Pile datasets.

Source: RWKV-7 https://arxiv.org/pdf/2503.14456
Source: Gated DeltaNet, https://arxiv.org/pdf/2412.06464

iv. Context retrieval performance (+ online learning): G-DeltaNet outperforms Mamba2 on long sequences.

Table 2: Zero-shot performance comparison on S-NIAH benchmark suite for 1.3B models (see §4 for setups)

	S-NIAH-1 (pass-key retrieval)				S-NIAH-2 (number in haystack)				S-NIAH-3 (uuid in haystack)		
Model	1K	2K	4K	8K	1K	2K	4K	8K	1K	2K	4K
DeltaNet	97.4	96.8	**99.0**	**98.8**	98.4	45.6	18.6	14.4	85.2	47.0	22.4
Mamba2	**99.2**	**98.8**	65.4	30.4	99.4	98.8	56.2	17.0	64.4	47.6	4.6
Gated DeltaNet	98.4	88.4	91.4	91.8	**100.0**	**99.8**	**92.2**	**29.6**	**86.6**	**84.2**	**27.6**

S-NIAH-3: uuid in a haystack

Context:

A special magic uuid is hidden within the following text. Make sure to memorize it. I will quiz you about the uuid afterwards.
What hard liquor, cigarettes, heroin, and crack have in common is that they're all more concentrated forms of less addictive predecessors. Most if not all the things we describe as addictive are. [....] One of the special magic uuid for vague-ecology is: 8a14be62-295b-4715-8333-e8615fb8d16c. And the scary thing is, the process that created them is accelerating. We wouldn't want to stop it. It's the same process that cures diseases: technological progress. Technological progress means making things do more of what we want. When the thing we want is something we want to want, we consider technological progress good [....]
Query: "What is the special magic uuid for vague-ecology?"
Expected answer: "8a14be62-295b-4715-8333-e8615fb8d16c"

1.5 Experimental Result

v. Scaling law: Hybrid linear and softmax attention can achieve GPT-4o level performance

MiniMax-01 (MiniMax et al. 2025) used

- Hybrid attention: 7/8 linear attention layers + 1/8 softmax attention layer
- Lightning attention (Qin et al. 2024b): simple linear attention with data-independent decay

Outline – Efficient Attention Variants

1. Linear Attention Machenism

- **Data-dependent decay:** RetNet, LighteningAttention, Mamba2, GLA
- **Test time online learning:** DeltaNet, Test-Time-Training, Titans, RWKV7, Gated DeltaNet

2. Sparse Attention Mechanisms

- **Static Sparsity:** BigBird, StreamingLLM, H2O
- **Dynamic Sparsity**: Native Sparse Attention (DeepSeek), MoBA (Kimi)

3. Hybrid Attention Mechanisms

- **Inter-layer mixing**: Minimax-01, Jamba, Samba
- **Intra-layer mixing:** Hymba

2. Sparse Attention Mechanisms: Static

Content-independent sparse patterns: BigBird, Window Attention, Streaming LLM, etc.

Content-dependent sparse patterns: H2O

Underlying structure of H2O

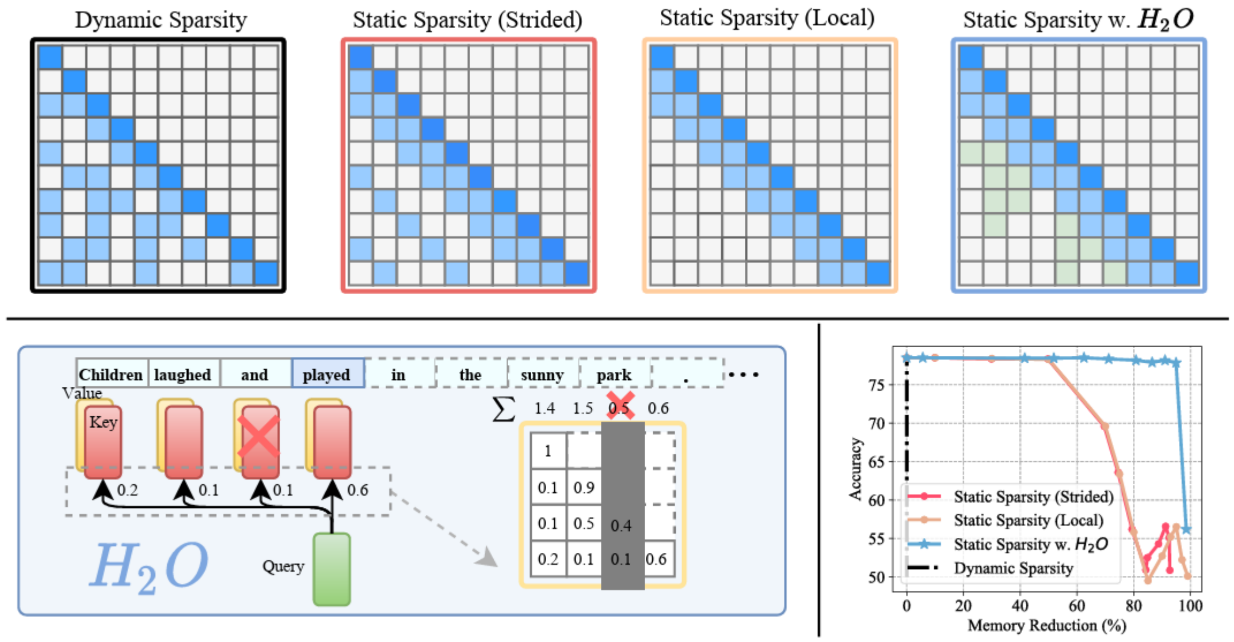


Figure 1: Upper plots illustrate symbolic plots of an attention map deploying different KV cache policies in LLM generation. Lower right: contrasts their accuracy-memory trade-off. Left: the overview of H_2O framework.

- Widely used in the early stages of LLM research, they can simultaneously reduce computational complexity and the storage size of KV Cache.
- Some information is permanently lost, which led to subsequent research on dynamic sparse attention.

H2O (NIPS23): https://arxiv.org/pdf/2306.14048
NSA (ACL25): https://arxiv.org/pdf/2502.11089

2. Sparse Attention Mechanisms: Dynamic

Dynamically determining the sparse pattern,

Challenge: How to maintain hardware efficiency?

- Discontinuous sparsity cannot achieve the theoretical sparsity speedup ratio.

Common solution: Selecting the top k key/value blocks for different queries.

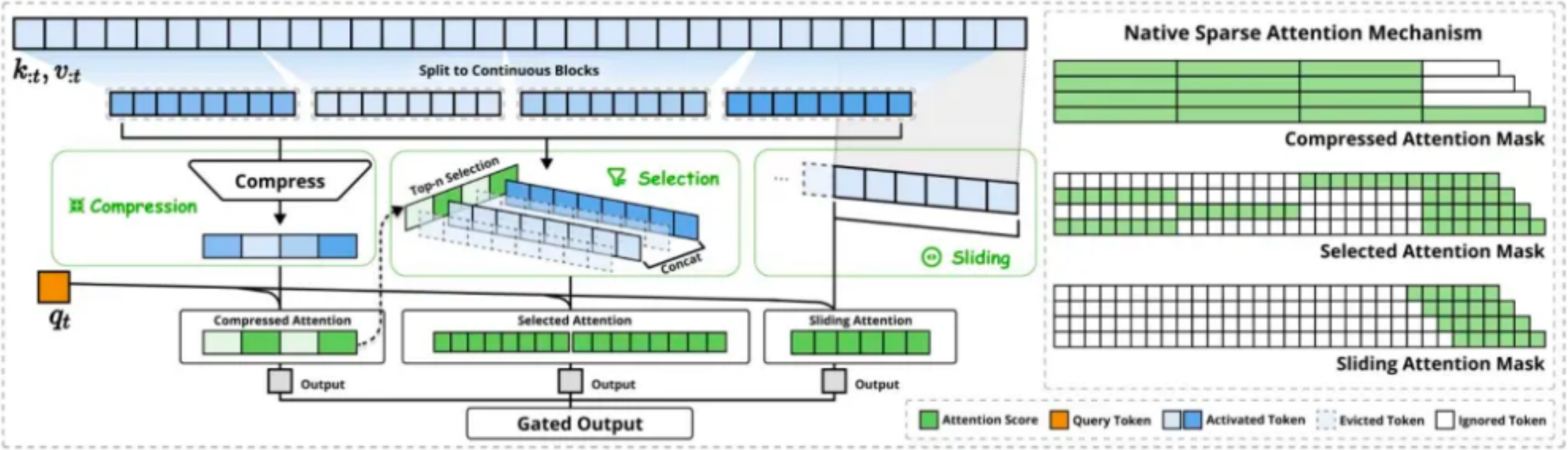
- Reading each block continuously can better align with hardware.

Works: Native Sparse Attention (DeepSeek), MoBA (Kimi)

H2O (NIPS23): https://arxiv.org/pdf/2306.14048
NSA (ACL25): https://arxiv.org/pdf/2502.11089

2. Sparse Attention Mechanisms: Dynamic (NSA)

Underlying structure of Native Sparse Attention (DeepSeek)



- Three branches: compression, block selection, sliding window
- Compression and block selection share attention scores; they can be pre-trained directly.
- Key assumption: Each head under each query group selects the same KV block, which avoids different heads repeatedly reading different KV blocks, thereby reducing I/O overhead.

H2O (NIPS23): https://arxiv.org/pdf/2306.14048
NSA (ACL25): https://arxiv.org/pdf/2502.11089

2. Sparse Attention Mechanism: Dynamic (MoBA)

Underlying Structure of MoBA (Kimi)

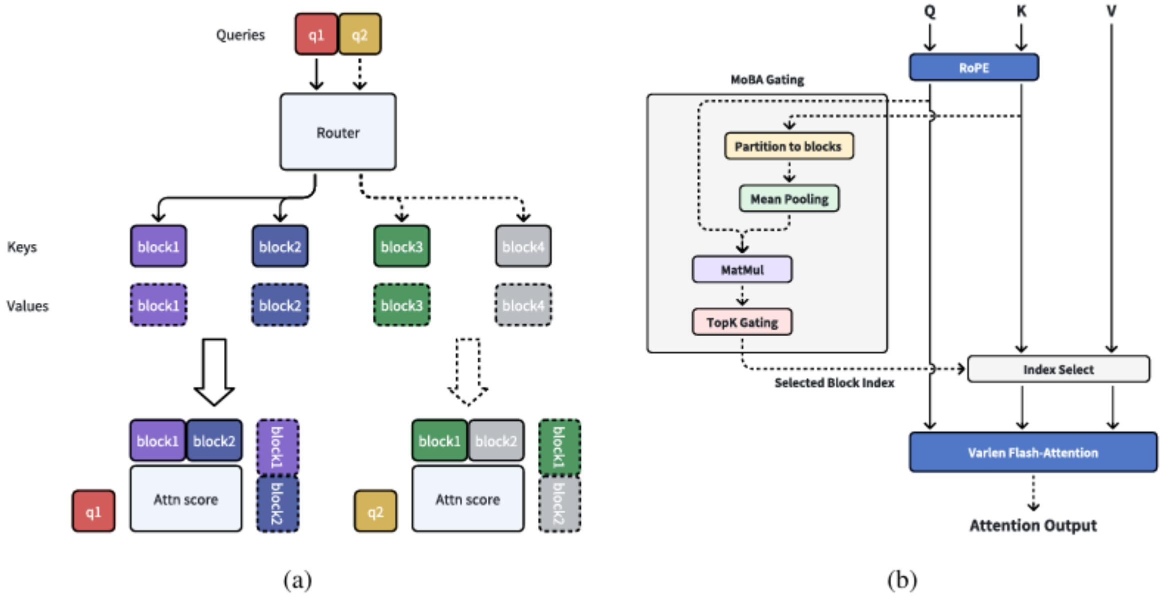


Figure 1: Illustration of mixture of block attention (MoBA). **(a)** A running example of MoBA; **(b)** Integration of MoBA into Flash Attention.

- Minima design, no new parameters introduced (mean pooling), no forced selection of neighboring blocks.
- Compatible with flash-attention kernel.

MoBA: *https://arxiv.org/pdf/2502.13189*

2. Sparse Attention Mechanism: Dynamic

Model	SQA			MQA				Synthetic		Code	Avg.
	MFQA-en	MFQA-zh	Qasper	HPQ	2Wiki	GovRpt	Dur	PassR-en	PassR-zh	LCC	
H2O	0.428	0.429	0.308	0.112	0.101	0.231	0.208	0.704	0.421	0.092	0.303
InfLLM	0.474	0.517	0.356	0.306	0.250	0.277	0.257	0.766	0.486	0.143	0.383
Quest	0.495	0.561	0.365	0.295	0.245	0.293	0.257	0.792	0.478	0.135	0.392
Exact-Top	0.502	0.605	0.397	0.321	0.288	0.316	0.291	0.810	0.548	0.156	0.423
Full Attn	**0.512**	0.623	0.409	0.350	0.305	**0.324**	0.294	0.830	**0.560**	0.163	0.437
NSA	0.503	**0.624**	**0.432**	**0.437**	**0.356**	0.307	**0.341**	**0.905**	0.550	**0.232**	**0.469**

Generation Token Limit	8192	16384
Full Attention-R	0.046	0.092
NSA-R	**0.121**	**0.146**

Table 3 | AIME Instruction-based Evaluating after supervised fine-tuning. Our NSA-R demonstrates better performance than Full Attention-R at both 8k and 16k sequence lengths

L(C)	MoBA	Full
LM loss (seqlen=8K)	$2.625 \times C^{-0.063}$	$2.622 \times C^{-0.063}$
Trailing LM loss (seqlen=32K, last 2K)	$1.546 \times C^{-0.108}$	$1.464 \times C^{-0.097}$

Experimental results for NSA and MoBA,

- NSA can even achieve better results than softmax attention;
- MoBA performs similarly to softmax attention.

Outline – Efficient Attention Variants

1. Linear Attention Machenism

- **Data-dependent decay:** RetNet, LighteningAttention, Mamba2, GLA
- **Test time online learning:** DeltaNet, Test-Time-Training, Titans, RWKV7, Gated DeltaNet

2. Sparse Attention Mechanisms

- **Static Sparsity:** BigBird, StreamingLLM, H2O
- **Dynamic Sparsity**: Native Sparse Attention (DeepSeek), MoBA (Kimi)

3. Hybrid Attention Mechanisms

- **Inter-layer mixing**: Minimax-01, Jamba, Samba
- **Intra-layer mixing:** Hymba

3. Mixed Attention Mechanism

Inter-layer mixing: different layers use different attention mechanisms
Example: Jamba, MiniMax-01 (MoE, 456B)

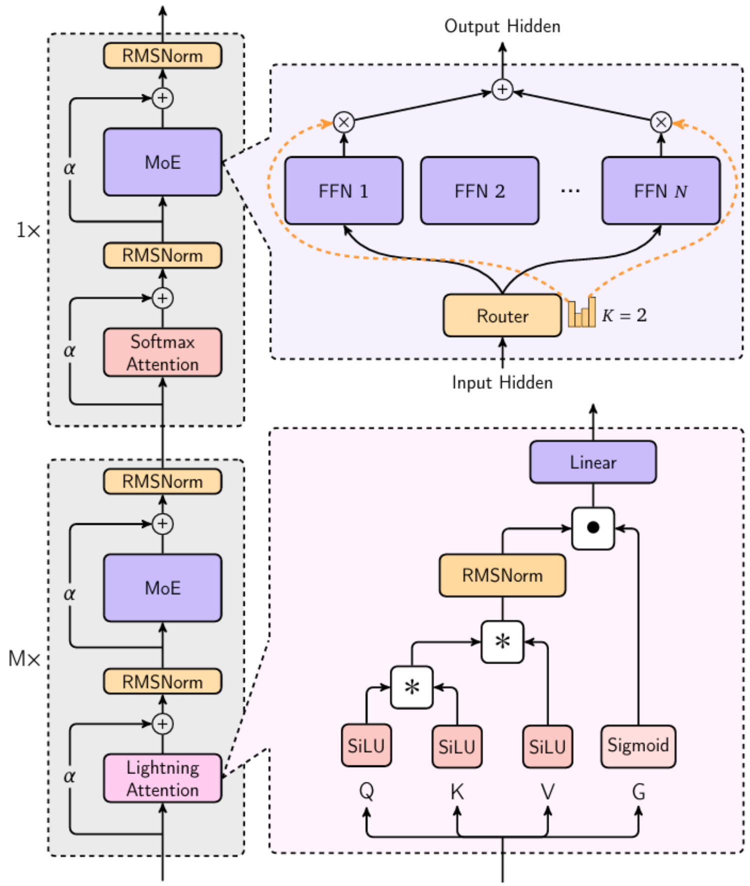


Figure 3 | **The architecture of MiniMax-Text-01.**

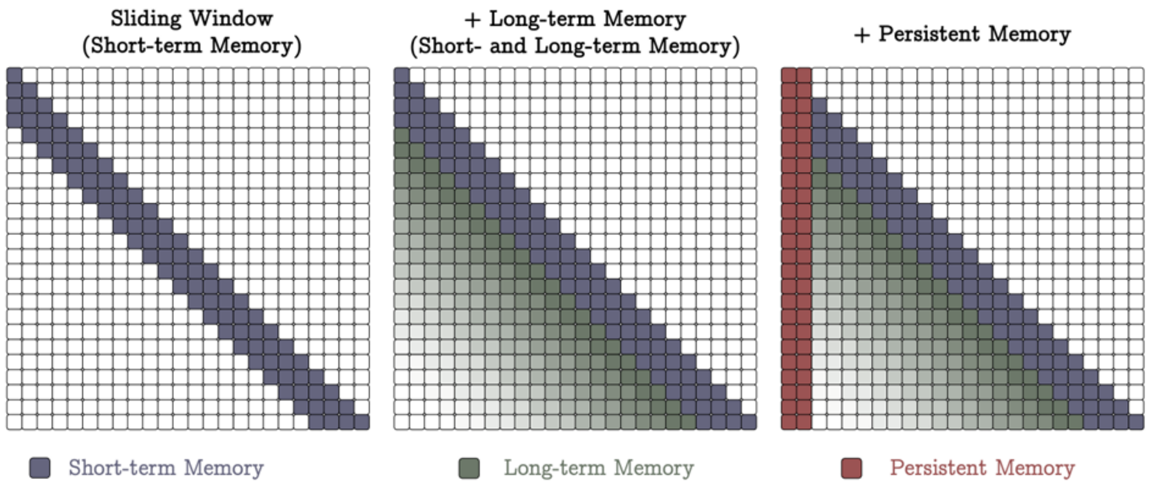
- Mixing ratio: 7 linear attention : 1 softmax attention;
- Since the KV cache for linear attention is negligible, it saves 7/8 of the KV cache across layers.

MiniMax01: https://arxiv.org/pdf/2501.08313
Titans: https://arxiv.org/pdf/2501.00663

3. Mixed Attention Mechanism

Intra-layer mixing: A single layer can have both sparse attention and linear attention
Example: Hymba

Intra-layer mixing: MAG variant of Titans.



(b) **Memory as Gating (MAG).** We use sliding window attention (SWA) as a short-term memory and our neural memory module as a long-term memory, combining by a gating.

- A single layer can have both sparse attention and linear attention
- Sliding window attention: Short-term memory
- Neural memory module: Long-term memory

MiniMax01: https://arxiv.org/pdf/2501.08313
Titans: https://arxiv.org/pdf/2501.00663

4. Summary of Efficient Attention Variants

	representative works	publish time	core improve	training			inference		
				seq length	model size	GPUs	seq lenth	task	
linear attention	Gated DeltaNet	MIT, 25.03	SGD loss for enhancing context retrieval	4k	0.1B / 0.4B / 1.5B / 2.9B	H100	20k	pass-key retrieval 1--3	100% / 91% / 42%
sparse attention	MoBA	KiMi, 25.02	Q & mean pooling K, choose topk index	32k	2.1 B	-	1 m (on continual training Lllama 8B)	catch up with Transformers; merged into product	
	NSA	DeepSeek, 25.02	compression, sliding window, and selection. optimized for hardware and pre-training	8k pretrain, 32k sft	3B	8GPU A100	4x faster for 8k; max 64k	pre-training, downstream evaluation, and reasoning: all better than full attention	
hybrid attention	nemotron-H	nvidia, 25.03	MAMMMM (M:Mamba2, A:Attn)	8192	8B / 56B	H100	60k input, 1024 output, 1.8X throughput	no better than qwen2.5 72B	
	minimax 01	minimax, 25.02	7linear:1 sofatmax, 80 layers	1 m	456B (MoE)	1500~2500 H800	-	similar to gpt-4o	
	qwen3	Ali, 25.05	gated delaNet+hybrid	32k	4B, 30B, 235B (MoE)	-	max 1m	SOTA open source thinking model	

Related Works in Huawei

Openings (Intern & Full-time):

zeng.wenqi@huawei.com

Published as a conference paper at ICLR 2025

ZETA: LEVERAGING Z-ORDER CURVES FOR EFFICIENT TOP-k ATTENTION

Qiuhao Zeng[♠] **Jerry Huang**[♡◇] **Peng Lu**[♡] **Gezheng Xu**[♠]
Boxing Chen[♣] **Charles Ling**[♠★] **Boyu Wang**[♠★*]
[♠]University of Western Ontario [♡]Université de Montréal [◇]Mila
[♣]Noah's Ark Lab [★]Vector Institute

ABSTRACT

Over recent years, the Transformer has become a fundamental building block for sequence modeling architectures. Yet at its core is the use of self-attention, whose memory and computational cost grow quadratically with the sequence length N, rendering it prohibitively expensive for long sequences. A promising approach is top-k attention, which selects only the k most relevant tokens and achieves performance comparable to vanilla self-attention while significantly reducing space and computational demands. However, causal masks require the current query token to only attend to past tokens, preventing existing top-k attention method from efficiently searching for the most relevant tokens in parallel, thereby limiting training efficiency. In this work, we propose ZETA, leveraging **Z**-Order Curves for **E**fficient **T**op-k **A**ttention, to enable parallel querying of past tokens for entire sequences. We first theoretically show that the choice of key and query dimensions involves a trade-off between the curse of dimensionality and the preservation of relative distances after projection. In light of this insight, we propose reducing the dimensionality of keys and queries in contrast to values and further leverage Z-order curves to map low-dimensional keys and queries into *one*-dimensional space, which permits parallel sorting, thereby largely improving the efficiency for top-k token selection. Experimental results demonstrate that ZETA matches the performance of standard attention on the synthetic MULTI-QUERY ASSOCIATIVE RECALL task and outperforms attention and its variants on LONG RANGE ARENA and WIKITEXT-103 language modeling.

Published as a conference paper at COLM 2025

RESONA: Improving Context Copying in Linear Recurrence Models with Retrieval

Xinyu Wang[♠*] **Linrui Ma**[♣*] **Jerry Huang**[♡◇*] **Peng Lu**[♡]
Prasanna Parthasarathi[♣] **Xiao-Wen Chang**[♠] **Boxing Chen**[♣] **Yufei Cui**[♣†]

[♠] McGill University [♣] Noah's Ark Lab, Montreal [♡] Université de Montréal [◇] Mila

Abstract

Recent shifts in the space of large language model (LLM) research have shown an increasing focus on novel architectures to compete with prototypical Transformer-based models that have long dominated this space. Linear recurrent models have proven to be a viable competitor due to their computational efficiency. However, such models still demonstrate a sizable gap compared to Transformers in terms of in-context learning among other tasks that require recalling information from a context. In this work, we introduce RESONA, a simple and scalable framework for augmenting linear recurrent models with retrieval. RESONA augments models with the ability to integrate retrieved information from the provided input context, enabling tailored behavior to diverse task requirements. Experiments on a variety of linear recurrent models demonstrate that RESONA-augmented models observe significant performance gains on a variety of synthetic as well as real-world natural language tasks, highlighting its ability to act as a general purpose method to improve the in-context learning and language modeling abilities of linear recurrent LLMs.

Thank you.

Bring digital to every person, home and organization for a fully connected, intelligent world.

HUAWEI