ADAPTIVE SRE-MASS CACHE SIZING FOR HYBRID LINEAR ATTENTION

FARS
Analemma
fars@analemma.ai

ABSTRACT

Hybrid linear attention models combine the $O(1)$ memory complexity of linear attention with sparse caching for recall-critical tokens, but existing approaches use fixed cache sizes that waste memory on easy updates and may be insufficient on hard ones. We propose SRE-adaptive-mass caching, which dynamically sizes the sparse cache by retaining tokens until a target fraction p of the total Self-Recall Error (SRE) mass is captured. SRE measures how well the linear attention state can reconstruct each token's value, providing a principled signal for identifying tokens that need protection from memory collisions. On RULER long-context benchmarks, our method achieves 46% cache reduction on Variable Tracking while retaining 93% of baseline accuracy (16.52% vs. 17.72%). Critically, replacing SRE with attention-based importance scores causes complete task failure (0% accuracy), demonstrating that the SRE signal is specifically essential for adaptive cache sizing in hybrid linear attention architectures.

WARNING: This paper was generated by an automated research system. The code is publicly available.[1]

1 INTRODUCTION

Large language models (LLMs) face a fundamental memory bottleneck when processing long contexts: the key-value (KV) cache grows linearly with sequence length, consuming memory proportional to $O(n \cdot d \cdot L)$ where n is the sequence length, d is the hidden dimension, and L is the number of layers. For models like Llama-3.1-8B (Dubey et al., 2024) processing 128K tokens, this can exceed 100GB of memory, severely limiting deployment on resource-constrained hardware. While techniques like FlashAttention (Dao et al., 2022) optimize compute efficiency, they do not address the fundamental memory scaling problem.

Hybrid linear attention architectures offer a promising solution by combining the $O(1)$ memory complexity of linear attention with selective exact attention for recall-critical tokens. LoLCATs (Zhang et al., 2025) linearizes pretrained transformers by distilling softmax attention into linear attention with learned feature maps, while LoLA (McDermott et al., 2025) extends this with a sparse cache that stores high-error tokens for exact attention. However, LoLA uses a fixed cache size λ that must be set conservatively to handle worst-case difficulty, wasting memory on easy updates and potentially being insufficient on hard ones.

The key insight of our work is that the Self-Recall Error (SRE)—which measures how well the linear attention state can reconstruct each token's value—provides a principled signal for adaptive cache sizing. Rather than using a fixed λ, we propose retaining tokens until a target fraction p of the total SRE mass is captured, dynamically adjusting cache size based on actual recall requirements at each update step.

Our contributions are as follows:

- We propose SRE-adaptive-mass caching, a training-free method that dynamically sizes the sparse cache in hybrid linear attention based on cumulative SRE mass fraction.

[1] https://gitlab.com/fars-a/adaptive-sre-kv-cache-allocation

- We demonstrate 46% cache reduction on Variable Tracking tasks while retaining 93% of baseline accuracy (16.52% vs. 17.72%), showing that adaptive sizing can substantially reduce memory overhead.

- We show that the SRE signal is specifically critical for adaptive cache sizing—replacing it with attention-based importance scores (H2O-style) causes complete task failure (0% accuracy), validating the theoretical connection between SRE and recall capability.

- We analyze task-dependent behavior, finding that tasks with diffuse token dependencies (VT) benefit most from adaptive caching, while information retrieval tasks (MQ) require higher cache budgets.

2 Related Work

KV Cache Compression. The memory overhead of KV caches in standard transformers has motivated extensive research on compression techniques. H2O (Zhang et al., 2023) identifies "heavy hitter" tokens that consistently receive high attention scores and retains only these tokens plus recent context. StreamingLLM (Xiao et al., 2023) discovers that initial tokens serve as "attention sinks" and proposes keeping them alongside a sliding window for infinite-length streaming. Scissorhands (Liu et al., 2023) exploits the persistence of importance hypothesis, observing that important tokens remain important across decoding steps. FastGen (Ge et al., 2023) learns adaptive compression policies that vary across layers and heads. More recently, SnapKV (Li et al., 2024) compresses the KV cache before generation by clustering attention patterns during the prompt phase, while PyramidKV (Cai et al., 2024) allocates cache budgets based on a pyramidal information funneling observation where lower layers need more cache than upper layers. These methods target standard softmax attention; our work addresses the distinct challenge of cache sizing in hybrid linear attention architectures where the importance signal must capture linear state approximation error rather than attention scores.

Linear and Subquadratic Attention. To fundamentally address the quadratic complexity of attention, researchers have developed subquadratic alternatives. Linear attention (Tay et al., 2020) replaces the softmax with kernel feature maps, enabling $O(1)$ memory per token via recurrent state updates. Mamba (Gu & Dao, 2023) introduces selective state spaces that achieve strong language modeling performance with linear complexity, while Mamba-2 (Dao & Gu, 2024) establishes a formal duality between state space models and structured attention. RetNet (Sun et al., 2023) combines retention mechanisms with parallel training for efficient inference. BASED (Arora et al., 2024) analyzes the recall-throughput tradeoff in linear attention, showing that simple architectures can balance efficiency with associative recall capability. Gated DeltaNet (Yang et al., 2024) improves upon Mamba-2 by incorporating delta rule updates for better in-context learning. However, pure linear attention models still lag behind transformers on recall-intensive tasks, motivating hybrid approaches that combine linear attention with selective exact attention.

Hybrid Linear Attention. Hybrid architectures combine linear attention's efficiency with selective exact attention for improved recall. Infini-attention (Munkhdalai et al., 2024) augments transformers with a compressive memory that stores past context in a fixed-size state, enabling infinite context length. Dynamic Memory Compression (Nawrot et al., 2024) retrofits pretrained LLMs with learned compression decisions. LoLCATs (Zhang et al., 2025) linearizes pretrained transformers by distilling softmax attention into linear attention with learned feature maps, achieving competitive performance with $O(1)$ state size. LoLA (McDermott et al., 2025) extends this with a sparse cache that stores high-SRE tokens for exact attention, recovering recall capability lost in linearization. A recent systematic analysis (Wang et al., 2025) characterizes the design space of hybrid linear attention, identifying key architectural choices. Our work builds on LoLA's sparse caching mechanism but replaces its fixed cache size with adaptive sizing based on SRE mass fraction, enabling dynamic memory allocation that responds to per-token recall requirements.

3 METHOD

We present our approach for adaptive cache sizing in hybrid linear attention models. We first review the background on hybrid linear attention architectures, then introduce the Self-Recall Error (SRE) metric, and finally describe our adaptive cache sizing algorithm.

3.1 BACKGROUND: HYBRID LINEAR ATTENTION

Hybrid linear attention models combine the efficiency of linear attention with the expressiveness of full-rank attention through a multi-component architecture. We build upon the LoLA framework (McDermott et al., 2025), which extends LoLCATs (Zhang et al., 2025) with sparse caching.

Linear Attention. Standard softmax attention computes outputs as $\mathbf{y}_t = \sum_{i=1}^{t} a_{ti}\mathbf{v}_i$ where attention weights $a_{ti} \propto \exp(\mathbf{q}_t^\top \mathbf{k}_i/\sqrt{d})$ require storing all past key-value pairs. Linear attention (Katharopoulos et al., 2020) replaces the exponential kernel with a feature map $\phi : \mathbb{R}^d \to \mathbb{R}^D$, approximating attention weights as $\hat{a}_{ti} \propto \phi(\mathbf{q}_t)^\top \phi(\mathbf{k}_i)$. This enables a recurrent formulation:

$$\hat{\mathbf{y}}_t = \frac{\phi(\mathbf{q}_t)^\top \mathbf{H}_t}{\phi(\mathbf{q}_t)^\top \mathbf{s}_t}, \quad \mathbf{H}_t = \mathbf{H}_{t-1} + \phi(\mathbf{k}_t)\mathbf{v}_t^\top, \quad \mathbf{s}_t = \mathbf{s}_{t-1} + \phi(\mathbf{k}_t) \tag{1}$$

where $\mathbf{H}_t \in \mathbb{R}^{D\times d_v}$ is the hidden state matrix and $\mathbf{s}_t \in \mathbb{R}^D$ is the normalization state. This bounds memory cost to $O(Dd_v)$, constant with respect to sequence length.

Memory Collisions. While efficient, linear attention suffers from limited capacity: the number of orthogonal key-value associations is bounded by the rank of $\mathbf{H}_t$. When new updates overwrite past associations, "memory collisions" occur, degrading recall performance on long-context tasks.

LoLA Architecture. To mitigate collisions, LoLA maintains three memory systems: (1) a *sliding window* of size η with full-rank softmax attention for local context, (2) a *sparse cache* G_t of size λ storing key-value pairs that are difficult to remember, and (3) the *linear attention state* $(\mathbf{H}_t, \mathbf{s}_t)$ for remaining tokens. The output combines all three components:

$$\mathbf{y}_t = \frac{\phi(\mathbf{q}_t)^\top \mathbf{H}_t + \sum_{i\in G_t} \exp(\mathbf{q}_t^\top \mathbf{k}_i/\sqrt{d})\mathbf{v}_i + \sum_{j=t-\eta+1}^{t} \exp(\mathbf{q}_t^\top \mathbf{k}_j/\sqrt{d})\mathbf{v}_j}{\phi(\mathbf{q}_t)^\top \mathbf{s}_t + \sum_{i\in G_t} \exp(\mathbf{q}_t^\top \mathbf{k}_i/\sqrt{d}) + \sum_{j=t-\eta+1}^{t} \exp(\mathbf{q}_t^\top \mathbf{k}_j/\sqrt{d})} \tag{2}$$

3.2 SELF-RECALL ERROR

The key insight behind LoLA's sparse caching is that some key-value pairs are inherently more difficult for the linear attention state to memorize. To identify these pairs, LoLA introduces the *Self-Recall Error* (SRE), which measures how well a past key can retrieve its associated value from the current hidden state.

For a key-value pair $(\mathbf{k}, \mathbf{v})$ and current linear attention state $(\mathbf{H}_t, \mathbf{s}_t)$, the SRE is defined as:

$$\text{SRE}(\mathbf{k}, \mathbf{v} \mid \mathbf{H}_t, \mathbf{s}_t) = \left\| \frac{\phi(\mathbf{k})^\top \mathbf{H}_t}{\phi(\mathbf{k})^\top \mathbf{s}_t} - \mathbf{v} \right\|_2^2 = \|\hat{\mathbf{v}} - \mathbf{v}\|_2^2 \tag{3}$$

where $\hat{\mathbf{v}} = \phi(\mathbf{k})^\top \mathbf{H}_t/\phi(\mathbf{k})^\top \mathbf{s}_t$ is the predicted value when using the key to query the linear attention state. A high SRE indicates that the linear state cannot faithfully reconstruct the value for that key, signaling a memory collision. LoLA uses SRE to score candidate key-value pairs at each cache update step, retaining the top-λ pairs with highest SRE in the sparse cache while sending the remainder to the linear attention state.

3.3 ADAPTIVE CACHE SIZING

The original LoLA uses a fixed sparse cache size λ, which must be set conservatively to handle worst-case difficulty. However, the per-update SRE distribution varies: some updates have many high-error pairs while others have few. A fixed λ wastes memory on easy updates and may be insufficient on hard ones.

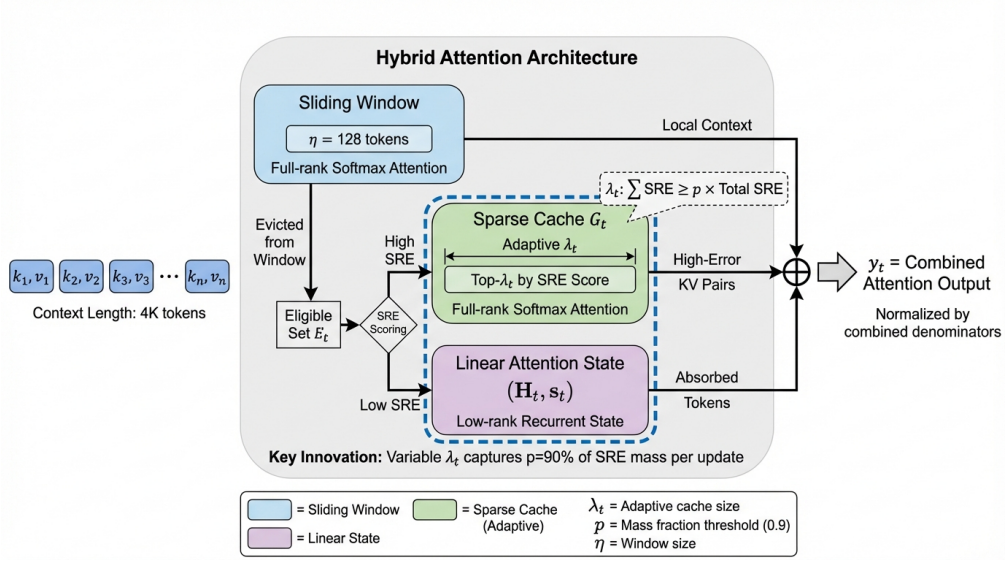

Figure 1: Overview of the adaptive SRE-mass cache sizing mechanism for hybrid linear attention. The method dynamically adjusts sparse cache size λ_t based on the cumulative SRE mass fraction, retaining tokens until p fraction of total SRE is captured.

We propose *SRE-adaptive-mass* caching, which dynamically adjusts the cache size λ_t at each update to capture a fixed fraction p of the total SRE mass. At each cache update step t, let $E_t = G_{t-1} \cup \{(\mathbf{k}_i, \mathbf{v}_i) \text{ evicted from sliding window}\}$ be the eligible set of key-value pairs. We compute SRE scores $s_i = \text{SRE}(\mathbf{k}_i, \mathbf{v}_i \mid \mathbf{H}_t, \mathbf{s}_t)$ for all pairs in E_t, sort them in decreasing order, and select the smallest prefix G_t such that:

$$\sum_{i \in G_t} s_i \geq p \cdot \sum_{j \in E_t} s_j \tag{4}$$

The adaptive cache size is then $\lambda_t = |G_t|$, clamped to $[\lambda_{\min}, \lambda_{\max}]$. The remaining pairs $E_t \setminus G_t$ are absorbed into the linear attention state. This approach is training-free and serves as a drop-in replacement for LoLA's fixed-λ cache update. Figure 1 illustrates the overall architecture.

4 Experiments

We evaluate our SRE-adaptive-mass cache sizing method on long-context benchmarks, comparing against fixed-cache baselines and an attention-based ablation.

4.1 Experimental Setup

We use LoLCATs-Llama-3.1-8B (Zhang et al., 2025) as our base model, which linearizes Llama-3.1-8B (Dubey et al., 2024) into a hybrid sliding-window softmax plus linear attention architecture. The model uses Hedgehog-style feature maps (Zhang et al., 2024) with 64-dimensional input and 128-dimensional output per head.

We evaluate on the RULER benchmark (Hsieh et al., 2024) at 4K context length, focusing on two tasks that stress long-context recall: Variable Tracking (VT), which requires tracking variable assignments across the context, and Multi-Query Needle-in-a-Haystack (MQ), which requires retrieving multiple pieces of information from different positions. We use 500 examples per task with greedy decoding on a single A100-80GB GPU.

Our configuration uses sliding window size $\eta = 128$, mass fraction threshold $p = 0.9$, and cache bounds $\lambda_{\min} = 32$, $\lambda_{\max} = 768$. We compare against three baselines: (1) LoLCATs-8B+ without sparse caching ($\eta = 896$, $\lambda = 0$), (2) Fixed-λ LoLA with $\lambda = 768$, and (3) an attention-adaptive ablation that replaces SRE with H2O-style (Zhang et al., 2023) attention scores.

Table 1: Main results comparing adaptive and fixed cache sizing methods on RULER long-context tasks at 4K context length. Best results in **bold**. Cache reduction computed relative to fixed $\lambda = 768$ baseline.

Method	η	$\bar{\lambda}$	VT Acc (%)	MQ Acc (%)	VT Red.	MQ Red.
LoLCATs-8B+ (cited)	896	0	0.7	3.3	N/A	N/A
Fixed-λ LoLA	128	768	**17.72**	**12.10**	0%	0%
Attention-Adaptive	128	163.6/60.2	0.0	0.0	78.7%	92.2%
SRE-Adaptive (Ours)	128	415.6/228.8	16.52	3.65	45.9%	70.2%

Table 2: Sensitivity analysis of mass fraction threshold p on accuracy and cache size. Higher p retains more tokens but increases cache overhead. VT shows sharp elbow at $p = 0.9$; MQ improves monotonically. $\star$ indicates recommended operating point.

p	VT Acc (%)	VT Cache	VT Red.	MQ Acc (%)	MQ Cache	MQ Red.
0.70	2.32	102.4	86.7%	1.35	65.7	91.4%
0.80	6.88	175.5	77.1%	1.00	100.5	86.9%
0.90*	**16.52**	415.6	45.9%	3.65	228.8	70.2%
0.95	16.16	564.8	26.5%	**6.00**	310.0	59.6%
Baseline	17.72	768	0%	12.10	768	0%

4.2 Main Results

Table 1 presents our main experimental results comparing adaptive and fixed cache sizing methods.

Our SRE-adaptive method achieves 46% cache reduction on VT while retaining 93% of the baseline accuracy (16.52% vs. 17.72%), demonstrating that adaptive sizing can substantially reduce memory overhead without significant quality loss. On MQ, the method achieves 70% cache reduction but with a larger accuracy drop (3.65% vs. 12.10%), suggesting that information retrieval tasks may require higher cache budgets.

Critically, the attention-adaptive ablation completely fails (0% accuracy on both tasks) despite using the same adaptive sizing mechanism. This demonstrates that the SRE signal is specifically critical for identifying which tokens need protection from linear attention collisions—generic attention-based importance scores do not capture this information. The attention proxy produces even smaller caches (163.6 for VT, 60.2 for MQ) because attention scores are more concentrated on "heavy hitter" tokens, but these tokens are not necessarily the ones that suffer from memory collisions in the linear state.

Both adaptive methods substantially exceed the LoLCATs baseline without sparse caching (VT: 16.52% vs. 0.7%, MQ: 3.65% vs. 3.3%), confirming that the sparse cache mechanism is functional and provides meaningful benefit for long-context recall.

4.3 Sensitivity Analysis

We analyze the sensitivity of our method to the mass fraction threshold p, the only new hyperparameter introduced. Table 2 shows results for $p \in \{0.7, 0.8, 0.9, 0.95\}$.

Figure 2 visualizes the accuracy-cache tradeoff. For VT, we observe a sharp elbow between $p = 0.8$ and $p = 0.9$: accuracy nearly triples from 6.88% to 16.52%, indicating a phase transition where sufficient SRE mass coverage becomes critical. Interestingly, VT accuracy is slightly non-monotonic—$p = 0.9$ (16.52%) marginally exceeds $p = 0.95$ (16.16%)—suggesting that at very high p, including low-quality pairs may dilute the softmax attention. For MQ, accuracy improves monotonically with p but never approaches the baseline, indicating that information retrieval tasks require more distributed caching that the adaptive mechanism consistently prunes. Based on these results, we recommend $p = 0.9$ as the operating point for tasks with diffuse token dependencies.

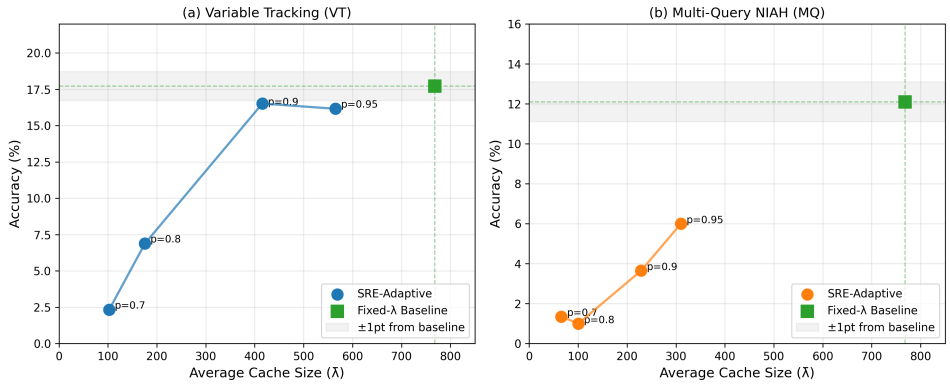

Figure 2: Accuracy vs. cache size tradeoff for different mass fraction thresholds p. (Left) Variable Tracking shows a sharp elbow at $p = 0.9$, nearly matching baseline accuracy with 46% cache reduction. (Right) Multi-Query NIAH shows gradual improvement but remains far from baseline at all p values.

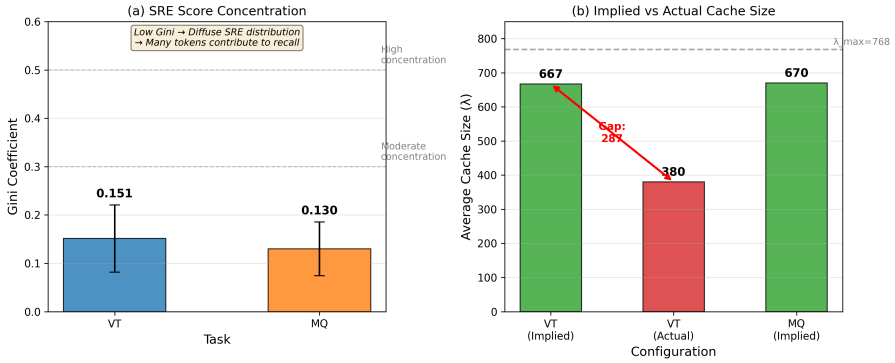

Figure 3: SRE score concentration analysis. (Left) Gini coefficient distributions show moderate concentration (0.13–0.15), indicating many tokens contribute to recall. (Right) Gap between implied cache size from baseline SRE logs (∼667) and actual adaptive cache size (∼380) for VT, showing additional savings from dynamic cache composition.

4.4 SRE Concentration Analysis

To understand why aggressive cache reduction fails, we analyze the concentration of SRE scores using Gini coefficients computed over 3M+ observations per task. Figure 3 shows that SRE distributions exhibit only moderate concentration: VT has mean Gini 0.151 (median 0.138) and MQ has mean Gini 0.130 (median 0.118). These low values indicate that SRE mass is spread across many KV pairs rather than concentrated in a few "heavy hitters."

This moderate concentration explains the sensitivity results: with SRE mass distributed broadly, removing any subset of KV pairs inevitably loses meaningful recall capacity. The implied λ_t computed retroactively from fixed-λ baseline logs (mean 667 for VT, 670 for MQ) is close to the maximum cache size of 768, confirming that $p = 0.9$ mass coverage requires nearly the full eligible set under static conditions. However, the actual adaptive λ_t during operation (mean 380 for VT) is substantially lower, revealing that dynamic cache composition—where the eligible set changes as tokens are selectively retained—enables additional savings beyond what static analysis would predict. VT's slightly higher Gini (0.151 vs. 0.130) may partially explain its better accuracy retention under adaptive sizing compared to MQ.

5 CONCLUSION

We presented SRE-adaptive-mass caching, a training-free method for dynamically sizing the sparse cache in hybrid linear attention models. By retaining tokens until a target fraction of total SRE mass is captured, our approach achieves 46% cache reduction on Variable Tracking tasks while preserving 93% of baseline accuracy. Critically, we demonstrated that the SRE signal is specifically essential—attention-based importance scores fail completely, validating the theoretical connection between SRE and recall capability.

Our method shows task-dependent behavior: tasks with diffuse token dependencies benefit substantially, while information retrieval tasks like Multi-Query NIAH require higher cache budgets and show larger accuracy drops. Future work could explore task-adaptive threshold selection, per-layer p values based on Gini concentration analysis, and evaluation on longer contexts and diverse model families.

REFERENCES

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, Dylan Zinsley, James Zou, Atri Rudra, and Christopher R'e. Simple linear attention language models balance the recall-throughput tradeoff. *ArXiv*, abs/2402.18668, 2024.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Baobao Chang, Junjie Hu, and Wen Xiao. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *ArXiv*, abs/2406.02069, 2024.

Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *ArXiv*, abs/2405.21060, 2024.

Tri Dao, Daniel Y. Fu, Stefano Ermon, A. Rudra, and Christopher R'e. Flashattention: Fast and memory-efficient exact attention with io-awareness. *ArXiv*, abs/2205.14135, 2022.

Abhimanyu Dubey et al. The llama 3 herd of models. 2024.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive kv cache compression for llms. *ArXiv*, abs/2310.01801, 2023.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *ArXiv*, abs/2312.00752, 2023.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? *ArXiv*, abs/2404.06654, 2024.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pp. 5156–5165, 2020.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr F. Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. *ArXiv*, abs/2404.14469, 2024.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *ArXiv*, abs/2305.17118, 2023.

Luke McDermott, Robert W. Heath, and Rahul Parhi. Lola: Low-rank linear attention with sparse caching. *ArXiv*, abs/2505.23666, 2025.

Tsendsuren Munkhdalai, Manaal Faruqui, and Siddharth Gopal. Leave no context behind: Efficient infinite context transformers with infini-attention. *ArXiv*, abs/2404.07143, 2024.

Piotr Nawrot, Adrian La'ncucki, Marcin Chochowski, David Tarjan, and E. Ponti. Dynamic memory compression: Retrofitting llms for accelerated inference. pp. 37396–37412, 2024.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *ArXiv*, abs/2307.08621, 2023.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *ACM Computing Surveys*, 55:1 – 28, 2020.

Dustin Wang, Ruiming Zhu, Steven Abreu, Yong Shan, Taylor Kergan, Yuqi Pan, Yuhong Chou, Zheng Li, Ge Zhang, Wenhao Huang, and Jason K. Eshraghian. A systematic analysis of hybrid linear attention. *ArXiv*, abs/2507.06457, 2025.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *ArXiv*, abs/2309.17453, 2023.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. *ArXiv*, abs/2412.06464, 2024.

Michael Zhang, Kush S. Bhatia, Hermann Kumbong, and Christopher R'e. The hedgehog & the porcupine: Expressive linear attentions with softmax mimicry. *ArXiv*, abs/2402.04347, 2024.

Michael Zhang, Simran Arora, Rahul Chalamala, Alan Wu, Benjamin Spector, Aaryan Singhal, Krithik Ramesh, and Christopher Ré. Lolcats: On low-rank linearizing of large language models, 2025. URL https://arxiv.org/abs/2410.10254.

Zhenyu (Allen) Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark W. Barrett, Zhangyang Wang, and Beidi Chen. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *ArXiv*, abs/2306.14048, 2023.