

Kwai Summary Attention

Chenglong Chu* Hao Peng* Hui Wang* Jiangxia Cao*[†]
Lu Ren* Zhiyuan Liang* Ziqi Wang* Hongtao Cheng
Yi Su Xinchun Luo Jian Liang Ruiming Tang[†]
Guorui Zhou Han Li
Kuaishou Technology

{chuchenglong, penghao06, wanghui28, caojiangxia, renlu05,
liangzhiyuan, wangziqi06, chenghongtao, suy10, luoxinchun,
liangjian03, tangruiming, zhouguorui, lihan08}@kuaishou.com

Abstract

Long-context capability has become increasingly critical for next-generation Large Language Models (LLMs). However, the quadratic time complexity of standard softmax attention introduces substantial computation and memory overhead as sequence length grows, limiting the scalability of LLMs in long-context scenarios. Beyond existing approaches such as linear recurrent architectures and KV-cache compression methods, we propose **Kwai Summary Attention (KSA)**, a novel attention mechanism that reduces sequence modeling cost by compressing historical contexts into learnable summary tokens. Specifically, summary tokens are periodically inserted into the input sequence and interleaved with text tokens, partitioning the sequence into distinct chunks while enabling text tokens and summary tokens to operate under different attention scopes. This design preserves dense local expressiveness through full attention among neighboring text tokens, while modeling long-range dependencies efficiently via summary tokens. Empirical results demonstrate that hybrid-KSA consistently outperforms existing hybrid attention variants in long-context modeling. With $k = 8$, each KSA layer compresses the sequence axis of its KV cache by $8\times$; our 3:1 hybrid achieves a measured $2.87\times$ KV-cache reduction at 128K while preserving model performance.

1 Introduction

In past years, the Transformer’s scaling-laws (Kaplan et al., 2020) achieve great success, transforming the landscape of deep learning. Excepted well-explored scaling directions of parameter and data,

the sequence modeling context length also provides a vital scaling perspective. With infrastructure improvement, valid context window gradually increases from earliest 1K to the latest 1M sequence (Qwen3.5, 2026; LongCat et al., 2026; Anthropic, 2026; Kimi2.6, 2026). As we know, the long-context capabilities of Large Language Models (LLMs) can significantly reduce hallucination and enable agent applications in solving intricate real-world tasks, such as Code Agent (Zhang et al., 2024), OpenClaw (OpenClaw, 2026) and so on.

In the agentic era, extensive memory across multi-turn conversations and accurate retrieval of historical contexts are prerequisites for highly intelligent agents and long chain-of-thought reasoning. However, vanilla attention (Vaswani et al., 2017) is subjected by its Achilles’ heel that imposes unaffordable storage and compute overhead as the sequence length grows: the KV cache scales linearly with sequence length, while the attention computation scales quadratically. To further unleash the long-context capability of LLMs and unlock the next tier of intelligence, inventing efficient attention variants and designing corresponding modeling architectures have become vital priorities.

Around the challenge of *reducing KV cache and computational overhead in long-context setting*, two technical paths have been well explored (more related works discussion in Appendix D):

- Reducing the KV cache per layer, evolving from MHA to GQA, *e.g.*, Qwen series (Qwen2 et al., 2024; Qwen3 et al., 2025; Qwen3.5, 2026), MLA (DeepSeek series (DeepSeek-V2 et al., 2024; DeepSeek-V3 et al., 2024), Kimi K2 (Kimi et al., 2025)), MLA-DSA (DeepSeek V3.2 (Liu et al., 2025),

*Contributed equally, listed alphabetically by first name.

[†]Corresponding Authors.

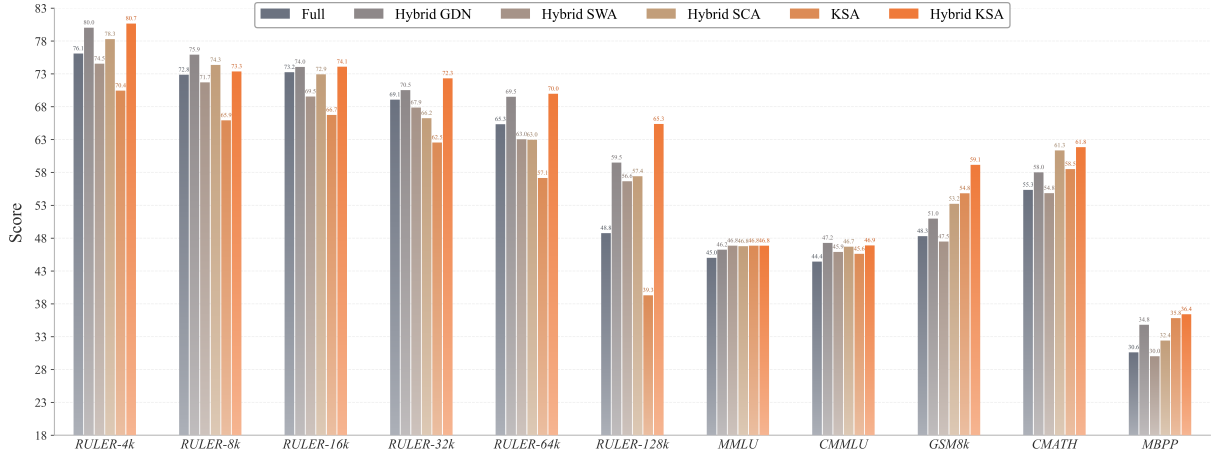


Figure 1: From scratch model variants performance on long-context and general benchmarks. Our hybrid-KSA ($k = 8$, corresponding to $8 \times$ sequence-axis compression within each KSA layer; 3:1 KSA/Full mixture ratio) leads on multiple long-context retrieval tasks and achieves competitive general understanding ability with other Hybrid variants.

GLM-5 (Zeng et al., 2026) and DeepSeek NSA (Yuan et al., 2025)), as refinement of the naive full attention. The core idea of this branch is to compress the per-token KV cache through head grouping and sharing, or low-rank KV projection. However, those efforts’ KV cache costs still maintain a strict 1:1 linear relationship with sequence length.

- Interleaving with KV Cache friendly architecture, with varying mixing ratios, represented by linear Gated DeltaNet + GQA (Qwen3.5, 2026) and SWA + GQA (Agarwal et al., 2025; Xiao et al., 2026). This branch replaces the majority of layers with efficient attention variants, fundamentally decoupling the KV cache from sequence length. However, the drawback is also obvious: for linear attention, the fixed-size state is inherently lossy compression which may blur long-range information; local variants, on the other hand, completely discard all contexts outside the window, thus losing perception of the distant information.

The above discussion reveals: i) KV cache compression methods still maintain strict linear relationship between sequence length and cache cost. ii) Architectures whose KV cache costs are independent of sequence length face limited modeling expressivity and impaired long-context capacity. However, we argue that there should be have a intermediate technique roadmap which has previously been underestimated: *Maintaining a linear relationship between the KV cache and sequence length, but performing semantic-level compression*

through a specific ratio k . This $O(n/k)$ path does not pursue a “minimum KV cache”, but rather trades acceptable memory costs for concrete and interpretable long distant dependency. Compared with SWA and linear attention, this design retains an explicit, albeit lossy, chunk-level representation of the distant history. **In other words, we believe that sequence-level token compression could provide a new perspective for reducing KV cache.** At the same time, DeepSeek V4 (DeepSeek-AI, 2026) model series have been released recently, whose design follows the first-principle of sequence-level KV Cache compression too, further proves the engineering feasibility and long-context robustness of this path.

In light of the above, we propose **Kwai Summary Attention (KSA)**, an efficient, adaptive, and scalable attention mechanism. KSA introduces a novel summary paradigm designed to distill historical contexts from lengthy sequences into a lightweight, learnable summary token. As illustrated in Figure 2a, our method operates by slicing the input sequence into multiple chunks of a specified size and injecting a learnable summary token at the end of each chunk. To achieve satisfactory trade-off between modeling capacity and efficiency in long-sequence processing, KSA enables text tokens and summary tokens to perform distinct computational patterns, as shown in Figure 2b: i) On the one hand, the summary tokens serve the purpose of distilling semantics in the current chunk, thus considering only text tokens within the same chunk. As the sequence grows, these tokens act as concise priors of distant contexts for subsequent text tokens.

ii) On the other hand, for the semantic fluency and stability, text tokens interact with neighboring text tokens via a sliding chunk mechanism, and could attend the long-range summary tokens of distant chunks. The local-text/global-summary visible tokens are able to provide favorable expressiveness and significantly reducing inference latency and KV-cache memory usage. To validate the effectiveness of KSA, we perform comprehensive from-scratch (Scratch) or continual pre-training (CPT) experiments on a wide range of downstream benchmarks and carry out detailed ablation studies with a series of architectural variants.¹

2 Methodology

2.1 Kwai Summary Attention

2.1.1 Summary Token Compression

Given an input sequence $\mathcal{T} = [t_0, \dots, t_{n-1}]$ of n text tokens and a chunk size k , we first partition it into n/k chunks (assume n is divisible by k) and append learnable summary token s_j at the end of each chunk j (all s_j share the special learnable summary token s). The augmented sequence $\hat{\mathcal{T}}$ is:

$$\begin{aligned} \hat{\mathcal{T}} &= [\text{chunk}_0, \dots, \text{chunk}_{\frac{n}{k}-1}], \\ \text{chunk}_j &= [t_{jk}, t_{jk+1}, \dots, t_{j(k-1)}, s_j]. \end{aligned} \quad (1)$$

where t_{jk} is the first text token of chunk j .

Moreover, we impose structural constraints on the information flow to make the visibility of summary tokens and text tokens complementary:

- Summary tokens can only see text tokens within their own chunk, and nothing else. Moreover, summary token position id is the same with its own chunk’s last text token’s.
- Text tokens can see sliding text tokens in neighbor chunks and distant summary tokens, without direct access to the full text history.

This design explicitly decouples token space from state space: *summary tokens focus on chunk semantic compression*, while *text tokens access long-range information through summary tokens*.

2.1.2 Sliding Chunk Attention

Besides summary token compression, we introduce a sliding chunk attention (SCA), with reasons analyzed in Appendix B.5. Formally, for a text token i

with chunk size k , SCA visible scope is:

$$[t_{(\lfloor \frac{i}{k} \rfloor - C)*k}, \dots, t_{i-1}, t_i] \quad (2)$$

where $\lfloor \cdot \rfloor$ is the floor division operator, and the $\lfloor \frac{i}{k} \rfloor - C$ indicate the starting chunk index.

KSA workflow combines the SCA and the summary tokens attention. Formally, for a text token i with chunk size k , its visible scope is:

$$\underbrace{[s_0, s_1, \dots, s_{\lfloor \frac{i}{k} \rfloor - C - 1}]}_{\text{distant summary tokens}} \cup \underbrace{[t_{(\lfloor \frac{i}{k} \rfloor - C)*k}, \dots, t_{i-1}, t_i]}_{\text{sliding chunk text tokens}} \quad (3)$$

The left term covers distant summaries beyond the window; the right term covers text tokens within the window. Note that summary tokens within the sliding chunk are *not* considered by text tokens.

Complexity. With C local chunks, KSA requires $\mathcal{O}(n^2/k + nCk + n)$ attention computation for training or prefill and $\mathcal{O}(n/k + Ck)$ cached token states for decoding. For fixed C and k , these remain quadratic in computation and linear in cache size, respectively; $1/k$ is a constant-factor sequence-axis compression rather than an asymptotic reduction.

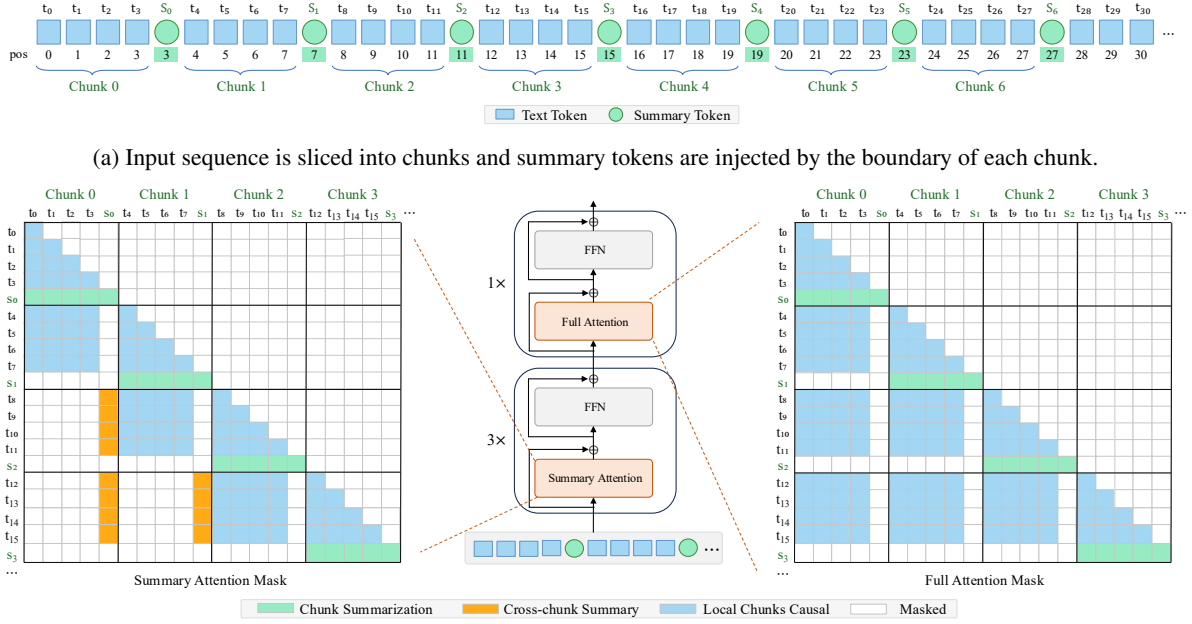
2.2 Kernel Design

The attention mask of KSA is a structured sparse mask defined as in Equation 3. With augmented input of length $L = n + n/k$ where n/k is the total number of summary tokens, building the dense mask would cost a prohibitive $\mathcal{O}(L^2)$ memory. To address this, we design a block-sparse attention kernel for training and prefill, and a summary KV cache for decoding, respectively. In the training / prefill kernel, Q/K/V are split into fixed-size blocks, and since KSA discard most Q-K interactions under sparse setting, the kernel only loads non-zero block pairs from HBM to SRAM for computation. We also provide a formal comparison of KSA’s cache efficiency with other attentions in Appendix E.

2.2.1 Efficient Decoding with Summary KV Cache

The nature of auto-regressive decoding requires every generation step to take all previous KV entries into account, resulting in a memory-bandwidth bottleneck. A naive approach of sparse attention requires concatenating, discarding, and re-allocating KV segments at every step, and these scattered memory operations hinder efficiency in decoding.

¹Resources: [code](#) and [kernel](#); [model weights](#).



(b) Summary tokens only see within chunk, while text tokens consider both local texts and distant summaries.

Figure 2: Our proposed **Kwai Summary Attention (KSA)** maintains quadratic complexity with an approximately $1/k$ reduction in the dominant long-range term while preserving expressivity in modeling long-range dependencies. By interleaving KSA with full attention blocks, hybrid-KSA achieves a satisfactory balance between performance and efficiency.

Composition of KSA KV cache (Figure 3a). We organize the KV cache as contiguous tensors, which can be logically divided into three distinct regions: i) *Current Chunk*, ii) *Sliding Chunk Text*, and iii) *Summary Token Buffer*. The *Current Chunk* region’s right edge always aligns to the *Sliding Chunk* region. This alignment ensures the caches are physically contiguous and can read all KV entries in a single memory operation. The *Summary Token Buffer* grows to the right as new summaries are generated. Since RoPE is applied before the cache, the physical layout does not interfere with each entry’s positional information.

Updating the current text chunk (Figure 3b). The KV states of incoming tokens are written sequentially into the *Current Chunk* region at the next available slot. During attention computation, the model retrieves the KV cache as a unified, contiguous slice that spans the current chunk, the sliding chunk text, and the distant summaries.

Chunk eviction and summary token insertion (Figures 3c, 3d, 3e). When the current chunk is filled, we update it by: i) The KV of the new summary token is first written into the left of *Current Chunk*, and it can be computed via a single memory-slicing operation (Figure 3c). ii) The just-finished

chunk text is then copied into the ring buffer and overwrites the oldest chunk in a modular fashion (Figure 3d). iii) Finally, the new summary KV is appended to the *Summary Token Buffer* (Figure 3e).

Conclusively, this design eliminates the need for other memory operations: *the cache layout itself inherently encodes the visibility rule*.

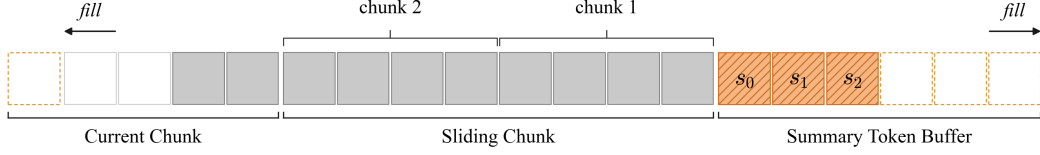
3 Experiments

3.1 Experiments Setting

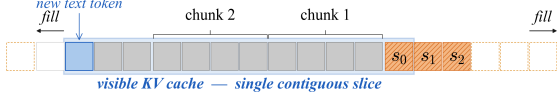
3.1.1 Model Configuration

We evaluate different attention architectures under two experimental settings: train-from-scratch (400B Token, 128K Sequence) and continual-pretraining (85B Token, 128K Sequence). The model architecture and training hyper-parameters are specified in Appendix B.1, and training recipes for different stages are in Appendix B.4.

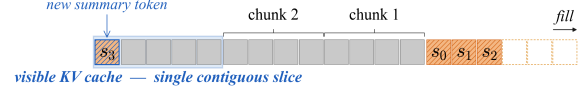
Baseline models. Our tested model variants are as follows: (1) **Pure Full Attention**, (2) **hybrid-GDN (Yang et al., 2025)/Ring-Linear (Ling-Team et al., 2025)**, (3) **Hybrid Sliding Window Attention (SWA)**, (4) **Hybrid Sliding Chunk Attention (SCA)**, (5) **Pure KSA**, (6) **hybrid-KSA**, all of the hybrid models with a 3:1 mixing ratio. Within each setting, variants share the original text-



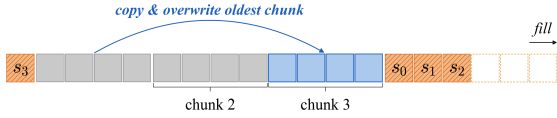
(a) The KSA KV cache consists of contiguous tensors with an organized layout, enabling reading KV states with only one memory operation without concatenation during decoding.



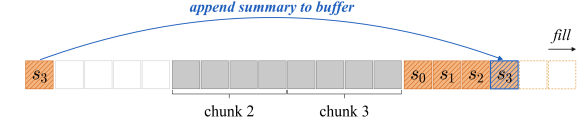
(b) Inserting a new text token: the KV is written into the next available slot of the current chunk, and attention reads a single contiguous slice.



(c) Inserting the chunk summary token: the self-KV is written into the scratch slot left of the current chunk, so summary attention is also a single contiguous slice.



(d) Replacing the oldest text chunk: the just-finished chunk is copied to the ring's write pointer, overwriting the oldest sliding chunk in place.



(e) Appending the summary token to the buffer: the new summary token is committed to the right end of the Summary Token Buffer.

Figure 3: The KSA cache features organized layout of contiguous tensors to enable efficient memory access. The lifecycle involves writing new tokens, inserting summaries, and recycling chunks in place.

token schedule and training hyperparameters.

Train-from-Scratch. We train a 1.9B-parameter language model from scratch. Training follows a progressive length schedule: the model is first trained on 250B tokens at a sequence length of 8K, followed by 50B tokens each at 32K, 64K, and 128K, respectively, adding up to 400B tokens.

Continual Pretraining (CPT). For continual pretraining, we initialize from Qwen3-4B-base, and train it on 85B tokens, with 25B, 35B, and 25B tokens at 32K, 64K, and 128K, respectively. Because Qwen3-4B-base uses GQA, all CPT variants retain its GQA configuration. Each configuration ran once on 256 NVIDIA H800 GPUs for about 60 hours, totaling about 337,920 H800 GPU-hours.

3.1.2 Datasets and Evaluation Benchmark

Training data. We train KSA on a diverse corpus consisting of Common Knowledge, Math, Code, STEM, General Reasoning, and Long Context. This wide coverage of training data preserves the model’s general knowledge, while boosting long-context capacity. The detailed distribution and pipeline are in Appendix B.2.

Evaluation benchmarks. For evaluation, we adopt RULER (Hsieh et al., 2024) for long-context, and incorporate multiple benchmarks across general knowledge, mathematics, and code to test

model’s real-world capability. We offer a concrete introduction of each benchmark in Appendix B.3.

3.2 CPT Performance Comparison

We evaluate the effectiveness of KSA under the CPT setting, comparing full **KSA** and **Hybrid-KSA** against four representative baselines: **Full** attention, sliding window attention (**Hybrid-SWA**), sliding chunk attention (**Hybrid-SCA**), and linear attention (**Hybrid-Linear**).

3.2.1 Long-Context Benchmark

We first examine KSA on RULER (Hsieh et al., 2024), a long-context stress test where it is designed to excel, against the introduced baselines. From Table 1, we can draw several conclusions:

i) *Hybrid-KSA demonstrates the strongest long-context retrieval capacity among all baselines.* Specifically, it achieves the best results at **RULER-4K (92.97)**, **RULER-32K (86.65)**, and **RULER-128K (71.67)**, outperforming the **Full** attention baseline with substantially lower cost.

ii) *Summary tokens effectively compress and propagate distant context information when full attention becomes prohibitive.* Notably, at the longest 128K context, **Hybrid-KSA** surpasses **Full** attention by **+5.81** points and outperforms the strongest hybrid baseline (**Hybrid-SWA**) by **+5.40** points.

iii) *KSA’s way of information aggregation is a*

Table 1: For CPT setting, **hybrid-KSA** achieves *superior long-context retrieval across all RULER lengths*, while both KSA variants closely match or exceed **Full** attention on knowledge, math, and coding benchmarks, establishing the *smallest performance gap among the evaluated efficient-attention alternatives*.

Benchmarks	Baselines				Ours	
	Full	Hybrid-SWA	Hybrid-SCA	Hybrid-Linear	KSA	Hybrid-KSA
Long-Context Retrieval						
RULER-4K	92.88	91.30	86.02	86.39	91.55	92.97
RULER-8K	91.38	88.03	84.28	83.86	86.78	90.53
RULER-16K	89.12	82.87	80.67	78.06	84.78	88.86
RULER-32K	84.74	78.94	76.89	76.48	80.30	86.65
RULER-64K	78.16	73.88	68.88	73.50	76.09	76.04
RULER-128K	65.86	66.27	60.94	67.98	66.81	71.67
General Knowledge						
MMLU	71.83	70.57	69.83	64.33	70.73	70.50
CMMLU	75.00	73.69	72.59	68.41	73.29	72.63
C-Eval	73.66	72.36	71.66	67.42	72.14	72.66
MMLU-Pro	46.36	45.23	45.11	38.83	45.70	45.39
Mathematics						
CMath	83.41	84.84	83.16	79.09	84.58	84.25
GSM8K	82.75	81.92	80.10	72.44	81.09	79.50
MATH	47.48	48.24	47.45	42.57	48.15	47.56
Code						
MBPP	61.30	61.70	59.60	55.30	61.50	62.20
HumanEval	58.54	61.89	61.89	54.58	60.97	62.50
Average	73.50	72.12	69.94	67.28	72.30	73.59

more faithful approximation of full attention than fixed-window or linearized alternatives. Compared to other hybrid variants (**SWA**, **SCA**, **Linear**), our **KSA** and **Hybrid-KSA** models consistently lead by clear margins across all RULER lengths.

3.2.2 General Benchmarks

To test if our method preserve model’s general ability on real-world problems, we further validate KSA across three knowledge domains: general knowledge (MMLU, CMMLU), mathematics (CMath, GSM8K), and code generation (MBPP). From Table 1, we observe the following:

- i) *CPT of summary attention preserves the pre-trained model’s general world knowledge.* On MMLU and CMMLU, the full **KSA** model achieves 70.73 and 73.29, closely matching the **Full** attention upper bound (71.83 / 75.00), and substantially outperforms **Hybrid-Linear** (64.33 / 68.41), which suffers from the limited representation capacity of fixed-size memory updates.
- ii) *KSA is capable of mathematical reasoning.* The **KSA** model attains 84.58 on CMath, surpassing **Full** attention (83.41), and remains competitive on GSM8K (81.09 vs. 82.75).
- iii) *Our method robustly scales to real-world coding problems.* **Hybrid-KSA** achieves the best MBPP score (62.20) across all configurations.

- iv) *The proposed approaches consistently yield smaller capability gaps relative to **Full** attention than the other evaluated efficient-attention baselines.*

3.3 From-scratch Performance Comparison

We further evaluate KSA in a train-from-scratch setting, which isolates the effect of pretrained model capabilities based on massive corpora. We compare **KSA** and **Hybrid-KSA** against **Full**, **Hybrid-SWA**, **Hybrid-SCA**, and **Hybrid-GDN**.

3.3.1 Long-Context Benchmark

Our methods showcase promising performance on the RULER benchmark as in Table 2:

- i) ***Hybrid-KSA** achieves the best overall performance, surpassing even **Full** attention by a large margin.* It achieves the best scores at **RULER-4K (80.65)**, **32K (72.30)**, **64K (69.95)**, and **128K (65.35)**. Notably, at **128K**, it surpasses **Full** attention by a large margin (+16.60).
- ii) *KSA significantly improves robustness at extreme context lengths.* While **Full** attention and other hybrid baselines degrade rapidly as sequence length increases: *e.g.*, **Full**: 76.08 $\rightarrow$ 48.75, **Hybrid-KSA** maintains reliably performing, *i.e.*, 80.65 $\rightarrow$ 65.35, indicating strong robustness.
- iii) *Summary-based aggregation provides a stronger alternative to existing efficient attention*

Table 2: **Hybrid-KSA** consistently outperforms full and efficient-attention baselines, achieving superior long-context scalability and strong general performance in from-scratch training.

Benchmarks	Baselines				Ours	
	Full	Hybrid-SWA	Hybrid-SCA	Hybrid-GDN	KSA	Hybrid-KSA
Long-Context Retrieval						
RULER-4K	76.08	74.54	77.72	79.83	70.44	80.65
RULER-8K	72.85	71.69	75.22	76.01	65.91	73.35
RULER-16K	73.24	69.54	72.55	74.04	66.74	74.07
RULER-32K	69.06	67.86	67.74	70.41	62.54	72.30
RULER-64K	65.32	63.03	63.54	69.39	57.13	69.95
RULER-128K	48.75	56.64	58.01	59.87	39.29	65.35
General Knowledge						
MMLU	44.99	46.84	46.77	46.23	46.83	46.83
CMMLU	44.41	45.89	46.42	47.19	45.59	46.88
C-Eval	44.28	43.54	47.62	45.54	45.69	44.13
MMLU-Pro	19.48	20.46	20.10	21.22	21.72	22.52
Mathematics						
CMath	55.33	54.83	62.33	58.00	58.50	61.83
GSM8K	48.29	47.46	52.39	50.95	54.81	59.14
MATH	23.38	31.46	28.82	33.30	30.04	36.92
Code						
MBPP	30.60	30.00	31.60	34.80	35.80	36.40
HumanEval	25.61	28.05	26.83	27.44	29.88	31.71
Average	49.44	50.12	51.84	52.95	48.73	54.80

designs. Compared to **Hybrid-SWA**, **Hybrid-SCA**, and **Hybrid-GDN**, **Hybrid-KSA** consistently scores higher, especially in the long-context regime, *e.g.*, +5.48 over **GDN** at 128K.

3.3.2 General Benchmarks

We next evaluate general capabilities across knowledge, mathematics, and code benchmarks:

i) *Our method maintains competitive general knowledge performance.* On MMLU and CMMLU, **Hybrid-KSA** achieves **46.83** and **46.88**, matching or exceeding most baselines and remaining close to the best-performing methods.

ii) *KSA yields strong gains in mathematical reasoning.* **Hybrid-KSA** achieves the top on GSM8K (**59.14**) and MATH (**36.92**), outperforming **Full** attention by **+10.85** and **+13.54**, respectively.

iii) *Summary attention improves performance on code generation tasks.* On MBPP and HumanEval, **Hybrid-KSA** achieves the highest scores (**36.40** and **31.71**), outperforming **Full** attention.

iv) *The proposed method provides a better efficiency–performance trade-off than prior efficient-attention methods.* Across all benchmarks, **Hybrid-KSA** consistently ranks among the top, while reducing the dominant long-range attention term by an approximately $1/k$ factor.

3.4 KSA Design Ablation Analyses

3.4.1 Hybrid-KSA Configuration Analysis

We conduct ablations on key design choices of Hybrid-KSA, including chunk number N , chunk size S , and the hybrid ratio between summary and full attention; and put the results in Table 3. We also carry out more ablation studies and explorations and put them in Appendix C.

Chunk Number (N). Varying N controls the granularity of summarization. As shown in Table 3, $N=128$ strikes a balance between long-context ability and general performance:

i) *Moderately increasing N improves long-context performance, but larger values yield diminishing or even negative returns.* The RULER average improves from **82.80** to **83.16** as N increases from 32 to 64. However, further increasing N can degrade performance at longer ranges: *e.g.*, RULER-64k: 76.35 at $N=128$ vs. 65.73 at $N=256$.

ii) *N has limited impact on general benchmarks.* Performance remains stable as N varies, with slight improvement at $N=256$: *i.e.*, reasoning avg: 75.89.

Chunk Size. Chunk size determines the receptive field within each segment. We observe a clear trade-off between long-context and general capability:

i) *Smaller chunks yield the stronger long-context performance.* For example, $S=8$ scores the best on RULER average (**82.97**) and 64K (**76.35**).

Table 3: The default setting achieves the best balance between long-context and general performance.

Configuration	Long-Context (RULER)						Knowledge			Reasoning & Code			
	4K	8K	16K	32K	64K	Avg.	MMLU	CMMLU	Avg.	GSM8K	CMath	MBPP	Avg.
<i>(a) Chunk Num (N)</i>													
N = 32	91.02	89.46	<u>84.42</u>	<u>78.35</u>	<u>70.74</u>	82.80	69.93	72.68	<u>71.30</u>	81.23	<u>83.59</u>	60.50	<u>75.11</u>
N = 64	93.36	<u>88.65</u>	87.21	76.91	69.69	83.16	70.43	72.27	71.35	80.13	83.00	<u>61.20</u>	74.78
N = 128 (Default)	88.69	88.01	83.62	78.19	76.35	82.97	<u>70.18</u>	72.16	71.17	80.75	82.48	60.20	74.48
N = 256	<u>92.05</u>	88.30	83.86	79.40	65.73	81.87	69.94	<u>72.53</u>	71.23	<u>81.16</u>	84.92	61.60	75.89
<i>(b) Chunk Size (S)</i>													
S = 8 (Default)	88.69	88.01	83.62	78.19	76.35	82.97	70.18	72.16	71.17	80.75	82.48	60.20	74.48
S = 16	<u>88.77</u>	83.78	<u>82.20</u>	<u>77.34</u>	71.97	80.82	69.75	72.50	<u>71.13</u>	<u>80.81</u>	83.59	<u>61.20</u>	<u>75.20</u>
S = 32	90.21	<u>84.66</u>	<u>80.26</u>	<u>77.31</u>	<u>75.09</u>	<u>81.50</u>	69.91	71.99	70.95	81.23	<u>83.25</u>	<u>61.20</u>	75.23
S = 64	86.50	<u>81.61</u>	78.12	72.63	70.09	77.79	69.83	<u>72.28</u>	71.05	80.48	81.84	62.10	74.80
<i>(c) Hybrid Ratio (Summary : Full)</i>													
1:1	84.50	82.85	82.16	74.76	69.36	78.72	69.39	72.61	71.00	81.09	84.50	60.40	75.33
3:1 (Default)	88.69	88.01	<u>83.62</u>	78.19	76.35	82.97	70.18	72.16	71.17	<u>80.75</u>	82.48	<u>60.20</u>	74.48
5:1	93.75	89.12	87.54	78.47	<u>70.31</u>	83.84	<u>69.47</u>	<u>72.55</u>	<u>71.01</u>	79.83	<u>83.41</u>	59.20	74.15
8:1	<u>90.80</u>	85.88	78.94	71.83	66.19	78.73	68.31	71.06	69.69	79.98	82.91	58.10	73.67

ii) *Larger chunks improve general performance at the cost of long-context retrieval.* For instance, $S=32$ attains the best reasoning average (**75.23**) and strong GSM8K performance (**81.23**), while its RULER average drops to 81.50.

iii) *The default choice $S=8$ provides a balanced anchor point, slightly favoring long-context modeling while retaining strong general capability.*

Hybrid Ratio. Hybrid ratio is crucial for model to leverage different architectures’ strengths:

i) *Increasing the proportion of summary attention improves long-context performance but weakens general abilities.* For example, moving from 3:1 to 5:1 improves the RULER average (**82.97** $\rightarrow$ **83.84**), but degrades reasoning (74.48 $\rightarrow$ 74.15) and code performance (MBPP: 60.20 $\rightarrow$ 59.20).

ii) *Increasing full attention enhances general performance at the cost of long-context retrieval.* For instance, a 1:1 ratio improves reasoning (CMath: **84.50**) but significantly reduces long-context performance (RULER avg: 78.72).

iii) *The default choice 3:1 provides a balanced trade-off, achieving strong long-context performance with competitive general capability.*

3.4.2 Inference KV Cache and Speed Analysis

To provide an intuitive understanding of KSA’s efficiency, we test the KV cache memory footprint across context lengths 16-128K and the decode throughput at 16K, and plot the results in Figure 4:

i) *Hybrid-KSA retains a substantially smaller cache footprint than Full attention* For $k = 8$ and a 3:1 KSA/Full mixture, the layer-weighted asymptotic reduction is $4/(1 + 3/8) = 2.91\times$ before accounting for the local window and imple-

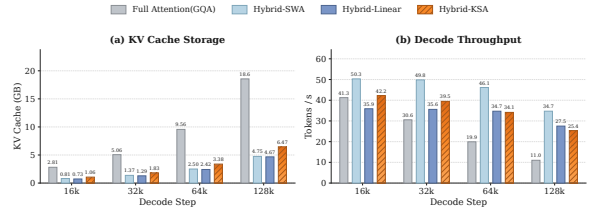


Figure 4: At 128K, Hybrid-KSA reduces the KV cache by $2.87\times$ relative to Full Attention while maintaining competitive or higher decode throughput compared with other efficient baselines.

mentation overhead. At 128K, **Hybrid-KSA** uses 6.47 GB versus 18.6 GB for **Full** attention, a measured **$2.87\times$ reduction**.

ii) *Our method achieves higher decode throughput than other efficient baselines.* At 16K, **Hybrid-KSA** achieves a throughput of $1.06\times$ relative to **Full** attention, surpassing both **Hybrid-SWA** and **Hybrid-Ring-Linear** which are $0.73\times$ and $0.81\times$ of **Full** attention respectively.

iii) These results demonstrate that KSA offers a favorable trade-off: *it compresses long-range context into a compact state, reclaiming memory without sacrificing decoding speed.*

4 Conclusion & Future Works

In this work, we reflect on long-context efficiency and highlight sequence-level KV cache compression: rather than pursuing either strict linear KV cache reduction, we advocate *sequence-level semantic compression* with a moderate ratio k , which maintains an $O(n/k)$ KV-cache dependence on sequence length while retaining explicit but lossy chunk-level representations of the distant history. Specifically, we propose **Kwai Summary Atten-**

tion (KSA), a drop-in efficient attention variant that i) periodically emits a learnable summary token for every text chunk, ii) routes local information through a sliding chunk and distant attention through summary tokens, and iii) guarantees a *non-overlapping information conflict*: every past chunk is either fully visible as raw text or accessed exclusively through its summary. Further, to support efficient training and inference, we contribute: i) a block-sparse training and prefill kernel that exploits the structured sparsity of KSA and avoids materializing the dense mask; ii) a contiguous, concatenation-free KV cache layout that encodes the visibility rule into memory allocation, thus every decoding step reads a single memory slice without further operation; and iii) a hybrid architectural recipe that interleaves full-attention and KSA layers to balance performance and cost. Across long-context retrieval, knowledge, mathematics, and coding, our **Hybrid-KSA** configuration consistently establishes the smallest quality gap to full attention among all baselines, while surpassing full attention on the most challenging tasks such as RULER-128K. These results support our motivation: *sequence-level token compression is empirically favorable for reducing KV cache size*.

In the future, we will explore the following directions: i) *Sparse summary attention*. Since summary tokens inherently encode chunk-level priors, we can further extend KSA by replacing the current full-attention layers with sparse attentions. In this design, summary tokens serve as routers that selectively activate attention over relevant text chunks, thereby avoiding the overhead of full attention. ii) *KSA post-training*. We currently only perform the pre-training process, and it is intriguing to explore KSA’s post-training, including supervised fine-tuning, reinforcement learning and on-policy distillation. iii) *Scaling laws of compression ratio*. The relationship between chunk size, performance, and task difficulty is not yet fully explored; establishing scaling laws for sequence-level compression, analogous to those for parameters and data, would guide the scaling of next level intelligence.

5 Limitations

KSA still has limitations. First, summary tokens introduce an explicit compression bottleneck, which may discard fine-grained token-level information required by exact retrieval or detailed reasoning tasks. Second, our experiments mainly focus on text-only LLMs; its effectiveness on multimodal

models and agentic workflows remains to be further studied. Third, the current hybrid-KSA design still relies on a small portion of full-attention layers, and the scaling laws of sequence-level compression are not yet fully understood. Finally, we do not report compute-matched MLA or MLA-DSA baselines because doing so would require retraining these architectures at our model sizes and token budgets; the KSA-MLA analysis in Appendix E.2 is theoretical only.

6 Ethical Considerations

This work proposes an efficient attention mechanism for large language models and does not introduce new datasets, collect personal information, or involve human subjects.

At the same time, KSA does not address broader risks associated with large language models, including hallucination, generation of harmful content, misinformation, privacy leakage, or misuse in downstream applications. Improving long-context capability may even amplify the impact of incorrect information if erroneous content is retained and propagated over longer contexts.

Therefore, KSA should be deployed together with existing safety, alignment, and monitoring mechanisms. We encourage future work to investigate the interaction between efficient long-context architectures and AI safety considerations.

References

- Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, and 1 others. 2025. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*.
- Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *EMNLP*, pages 4895–4901.
- Anthropic. 2026. [Introducing claude opus 4.6](#).
- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021. Program synthesis with large language models. *arXiv e-prints*, pages arXiv-2108.
- Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

- Aleksandar Botev, Soham De, Samuel L Smith, Anushan Fernando, George-Cristian Muraru, Ruba Haroun, Leonard Berrada, Razvan Pascanu, Pier Giuseppe Sessa, Robert Dadashi, and 1 others. 2024. Recurrentgemma: Moving past transformers for efficient open language models. *arXiv preprint arXiv:2404.07839*.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.
- Shaoshen Chen, Yangning Li, Zishan Xu, Yongqin Zeng, Shunlong Wu, Xinshuo Hu, Zifei Shan, Xin Su, Jiwei Tang, Yinghui Li, and Hai-Tao Zheng. 2025. DAST: Context-aware compression in LLMs via dynamic allocation of soft tokens. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20544–20552.
- Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. In *EMNLP*.
- Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- DeepSeek-AI. 2026. Deepseek-v4: Towards highly efficient million-token context intelligence.
- DeepSeek-V2, Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Daya Guo, and 1 others. 2024. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*.
- DeepSeek-V3, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.
- Chenlong Deng, Zhisong Zhang, Kelong Mao, Shuaiyi Li, Tianqing Fang, Hongming Zhang, Haitao Mi, Dong Yu, and Zhicheng Dou. 2025. UniGist: Towards general and hardware-aligned sequence-level long context compression. In *NeurIPS*.
- Xin Dong, Yonggan Fu, Shizhe Diao, Wonmin Byeon, ZIJIA CHEN, Ameya Sunil Mahabaleshwarkar, Shih-Yang Liu, Matthijs Van keirsbilck, Min-Hung Chen, Yoshi Suhara, Yingyan Celine Lin, Jan Kautz, and Pavlo Molchanov. 2025. Hymba: A hybrid-head architecture for small language models. In *ICLR*.
- Daniel Y Fu, Tri Dao, Khaled Kamal Saab, Armin W Thomas, Atri Rudra, and Christopher Re. 2023. Hungry hungry hippos: Towards language modeling with state space models. In *ICLR*.
- Gemma2, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, and 1 others. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.
- Paolo Gloriosi, Quentin Anthony, Yury Tokpanov, Anna Golubeva, Vasudev Shyam, James Whittington, Jonathan Pilault, and Beren Millidge. 2024a. The zamba2 suite: Technical report. *arXiv preprint arXiv:2411.15242*.
- Paolo Gloriosi, Quentin Anthony, Yury Tokpanov, James Whittington, Jonathan Pilault, Adam Ibrahim, and Beren Millidge. 2024b. Zamba: A compact 7b ssm hybrid model. *arXiv preprint arXiv:2405.16712*.
- Albert Gu and Tri Dao. 2024. Mamba: Linear-time sequence modeling with selective state spaces. In *CoLM*.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. In *ICLR*.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the math dataset. *NeurIPS*.
- Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. 2024. Ruler: What’s the real context size of your long-context language models? In *CoLM*.
- Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Yao Fu, and 1 others. 2023. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *Advances in neural information processing systems*, 36:62991–63010.
- Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, and Devendra Singh Chaplot. 2023. Mistral 7b. *Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, L  lio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timoth  e Lacroix, and William El Sayed*, pages 50–72.
- Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

- Kimi, Yifan Bai, Yiping Bao, Y Charles, Cheng Chen, Guanduo Chen, Haiting Chen, Huarong Chen, Jiahao Chen, Ningxin Chen, and 1 others. 2025. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*.
- Kimi2.6. 2026. [Kimi k2.6: Advancing open-source coding](#).
- Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. 2024. Cmmmlu: Measuring massive multitask language understanding in chinese. In *ACL*, pages 11260–11285.
- Yangning Li, Shaoshen Chen, Yinghui Li, Yankai Chen, Hai-Tao Zheng, Hui Wang, Wenhao Jiang, and Philip S. Yu. 2025. AdmTree: Compressing lengthy context with adaptive semantic trees. In *NeurIPS*.
- Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, and 1 others. 2024. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*.
- Ling-Team, Bin Han, Caizhi Tang, Chen Liang, Donghao Zhang, Fan Yuan, Feng Zhu, Jie Gao, Jingyu Hu, Longfei Li, and 1 others. 2025. Every attention matters: An efficient hybrid architecture for long-context reasoning. *arXiv preprint arXiv:2510.19338*.
- Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, and 1 others. 2025. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*.
- Meituan LongCat, Anchun Gui, Bei Li, Bingyang Tao, Bole Zhou, Borun Chen, Chao Zhang, Chen Gao, Chen Zhang, Chengcheng Han, and 1 others. 2026. Longcat-flash-thinking-2601 technical report. *arXiv preprint arXiv:2601.16725*.
- Lance Martin, Kedar Chandrayan, Gregory Kamradt, Lazaro Hurtado, Arkady Arkhangorodsky, Ikko Elto-ciear Ashimine, Prabha Arivalagan, and Pavel Král. 2023. [Needle in a haystack - pressure testing llms](#). *GitHub repository*.
- Jesse Mu, Xiang Lisa Li, and Noah Goodman. 2023. Learning to compress prompts with Gist Tokens. In *NeurIPS*.
- OpenClaw. 2026. [Openclaw — personal ai assistant](#). *GitHub repository*.
- Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, and 1 others. 2024. The fineweb datasets: Decanting the web for the finest text data at scale. In *NeurIPS*, volume 37, pages 30811–30849.
- Bo Peng, Daniel Goldstein, Quentin Gregory Anthony, Alon Albalak, Eric Alcaide, Stella Biderman, Eugene Cheah, Teddy Ferdinan, Kranthi Kiran GV, Haowen Hou, Satyapriya Krishna, Ronald McClelland Jr., Niklas Muennighoff, Fares Obeid, Atsushi Saito, Guangyu Song, Haoqin Tu, Ruichong Zhang, Bingchen Zhao, and 3 others. 2024. Eagle and finch: RWKV with matrix-valued states and dynamic recurrence. In *CoLM*.
- Qwen2, An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, and 44 others. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.
- Qwen3, An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, and 42 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.
- Qwen3.5. 2026. [Qwen3.5: Accelerating productivity with native multimodal agents](#).
- Noam Shazeer. 2019. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*.
- Dan Su, Kezhi Kong, Ying Lin, Joseph Jennings, Brandon Norick, Markus Kliegl, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. 2025. Nemotron-cc: Transforming common crawl into a refined long-horizon pretraining dataset. In *ACL*, pages 2459–2475.
- Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. 2023. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*.
- Yutao Sun, Li Dong, Yi Zhu, Shaohan Huang, Wenhui Wang, Shuming Ma, Quanlu Zhang, Jianyong Wang, and Furu Wei. 2024. You only cache once: Decoder-decoder architectures for language models. In *NeurIPS*.
- Jamba Team, Barak Lenz, Alan Arazi, Amir Bergman, Avshalom Manevich, Barak Peleg, Ben Aviram, Chen Almagor, Clara Fridman, Dan Padnos, and 1 others. 2024. Jamba-1.5: Hybrid transformer-mamba models at scale. *arXiv preprint arXiv:2408.12570*.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*, volume 30.
- Roger Waleffe, Wonmin Byeon, Duncan Riach, Brandon Norick, Vijay Korthikanti, Tri Dao, Albert Gu, Ali Hatamizadeh, Sudhakar Singh, Deepak

- Narayanan, and 1 others. 2024. An empirical study of mamba-based language models. *arXiv preprint arXiv:2406.07887*.
- Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyang Jiang, and 1 others. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290.
- Tianwen Wei, Jian Luan, Wei Liu, Shuang Dong, and Bin Wang. 2023. Cmath: Can your language model pass chinese elementary school math test? *arXiv e-prints*, pages arXiv–2306.
- Bangjun Xiao, Bingquan Xia, Bo Yang, Bofei Gao, Bowen Shen, Chen Zhang, Chenhong He, Chiheng Lou, Fuli Luo, Gang Wang, and 1 others. 2026. Mimo-v2-flash technical report. *arXiv preprint arXiv:2601.02780*.
- Shuai Yang, Wei Huang, Ruihang Chu, Yicheng Xiao, Yuyang Zhao, Xianbang Wang, Muiyang Li, Enze Xie, Ying-Cong Chen, Yao Lu, Song Han, and Yukang Chen. 2026. Longlive: Real-time interactive long video generation. In *ICLR*.
- Songlin Yang, Jan Kautz, and Ali Hatamizadeh. 2025. Gated delta networks: Improving mamba2 with delta rule. In *ICLR*.
- Tianyu Yu, Zefan Wang, Chongyi Wang, Fuwei Huang, Wenshuo Ma, Zhihui He, Tianchi Cai, Weize Chen, Yuxiang Huang, Yuanqian Zhao, and 1 others. 2025. Minicpm-v 4.5: Cooking efficient mlms via architecture, data, and training recipe. *arXiv preprint arXiv:2509.18154*.
- Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, and 1 others. 2025. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In *ACL*, pages 23078–23097.
- Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, and 1 others. 2026. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*.
- Jintao Zhang, Chendong Xiang, Haofeng Huang, Jia Wei, Haocheng Xi, Jun Zhu, and Jianfei Chen. 2025a. Spargeattn: Accurate sparse attention accelerating any model inference. In *ICML*.
- Kechi Zhang, Jia Li, Ge Li, Xianjie Shi, and Zhi Jin. 2024. Codeagent: Enhancing code generation with tool-integrated agent systems for real-world repo-level coding challenges. In *ACL*, pages 13643–13658.
- Peitian Zhang, Zheng Liu, Shitao Xiao, Ninglu Shao, Qiwei Ye, and Zhicheng Dou. 2025b. Long context compression with Activation Beacon. In *ICLR*.

A Overview

This appendix provides additional details and analyses that complement the main paper.

Section B presents the model configurations, training hyper-parameters, data pipeline, evaluation benchmarks, and training recipes used in our experiments.

Section B.5 discusses the motivation behind Sliding Chunk Attention (SCA) and explains why chunk-level routing is preferred over token-level sliding windows in the KSA framework.

Section C reports additional empirical results, including RULER subtask evaluations, Needle-in-a-Haystack retrieval experiments, training dynamics, and per-layer attention pattern analyses.

Section D reviews recent advances in efficient attention mechanisms and hybrid long-context architectures.

Finally, Section E provides a detailed theoretical analysis of KV-cache complexity and discusses KSA’s theoretical compatibility with GQA and MLA.

B Experiment Configuration

B.1 Model Setting and Hyper-Parameters

Table 4 and Table 5 show the model architecture and training hyper-parameters for our experiments.

B.2 Datasets Pipeline

The pretraining corpus is composed of six categories: Common Knowledge, Math, Code, STEM, General Reasoning, and Long Context. A comprehensive breakdown of data formulation is in Figure 5.

Long Context is a primary data class, as it directly serves our goal of reinforcing the model’s long-sequence modeling capacity. We construct a hybrid mixture tailored for context-length extension by combining naturally long documents, *i.e.*, books, long-form QA across code, math, and STEM; with synthetic sequences that probe long-range information tracking.

Common Knowledge forms the foundational layer, sourced from diverse high-quality web text (Penedo et al., 2024), educational content, and curated open-source corpora. The sources span filtered CommonCrawl snapshots (Su et al., 2025), web-extracted educational material, and high-quality open-domain text, ensuring broad coverage of general world knowledge.

Table 4: Training model configurations of our KSA.

Configuration	From Scratch	Continual Pretraining
Number of layers	24	36
Hidden size	2048	2560
Intermediate size	6144	9728
Attention heads (Q/KV)	16/16	32/8
Head dimension	128	128
Hybrid architecture ratio (KSA:Full)	3:1	3:1
Summary chunk size	8	8
Sliding chunk number	128	128
Tied embeddings	False	True

Table 5: Training hyperparameter settings.

Configuration	From Scratch	Continual Pretraining
Sequence Length Stages	8K / 32K / 64K / 128K	32K / 64K / 128K
Token Budget per Stage	250B / 50B / 50B / 50B	25B / 35B / 25B
Max Learning Rate	8K: 4×10^{-4} ; ≥ 32 K: 1×10^{-5}	All stages: 1×10^{-4}
Min Learning Rate		1×10^{-7}
RoPE Theta	8K: 10^4 ; ≥ 32 K: 10^6	10^6
Optimizer	AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$)	
Weight Decay	0.01	
LR Schedule	WSD (Yu et al., 2025)	
Gradient Clipping	1.0	

Math and **Code** are constructed from both natural and synthetic sources. The math subset integrates web-crawled mathematical content, competition problems, Chinese-language resources, and QA pairs, augmented with synthetic problem, *i.e.*, solution sequences to improve coverage of rare reasoning patterns. The code subset combines GitHub repositories, competition solutions, code-related QA pairs, and synthetic generation tasks spanning multiple programming languages.

STEM and **General Reasoning** broaden the model’s domain expertise through academic papers, textbook-style content, encyclopedia entries, long-form books, and structured QA corpora. Domain-specific scientific documents and exam-style data further strengthen coverage of technical disciplines.

The training pipeline follows the data mixture ratios shown in Figure 5 and progressively extends the context length across stages. In train-from-scratch experiments, the context length increases from 8K to 32K, 64K, and finally 128K tokens. In CPT experiments, training starts directly at 32K and advances to 128K, with all stages using the same ratios of data classes mixture.

B.3 Evaluation Benchmarks

We categorize the evaluation benchmarks into two groups, targeting long-context modeling ability and general language capability accordingly.

Long-Context Benchmarks: RULER (Hsieh et al., 2024) is a comprehensive long-context evaluation suite that assesses models across multiple dimensions including retrieval, multi-hop tracing, aggregation, and question answering, with configurable context lengths up to 128K tokens.

General Capability Benchmarks: We evaluate general capabilities across three domains: general knowledge, mathematics, and code. For general knowledge, MMLU (Hendrycks et al., 2021a) and its extensions, CMMLU (Li et al., 2024), C-Eval (Huang et al., 2023), and MMLU-Pro (Wang et al., 2024), assess knowledge breadth and reasoning across diverse domains via multi-choice questions, where CMMLU and C-Eval provide Chinese benchmarks spanning multiple disciplines and exam-style tasks, and MMLU-Pro introduces more challenging, professionally curated questions with reduced data contamination. For mathematics, GSM8K (Cobbe et al., 2021), CMATH (Wei et al., 2023), and MATH (Hendrycks et al., 2021b) cover problems from grade-school multi-step arith-

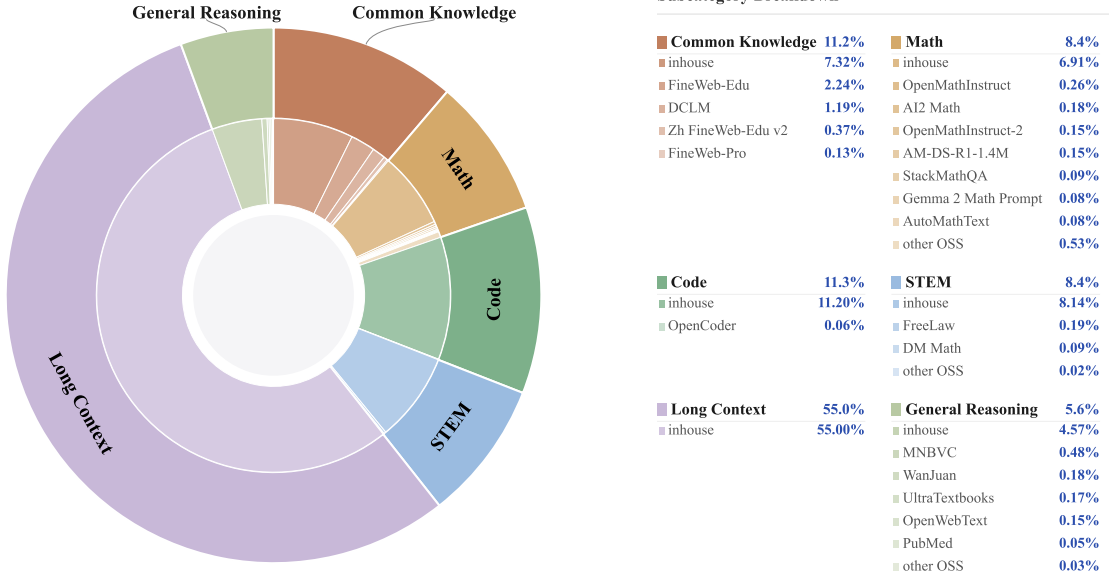


Figure 5: Overall Distribution of Pretraining Data Proportions.

metic to competition-level symbolic reasoning, including Chinese settings and diverse problem formats requiring multi-step derivations. For code, MBPP (Austin et al., 2021) and HumanEval (Chen et al., 2021) evaluate program synthesis from natural language, with HumanEval further emphasizing functional correctness through unit tests on more complex and realistic coding tasks.

B.4 Details of Training Recipe

To adapt summary attention with existing architecture seamlessly, we devise a three stage training stages for CPT: *summary token adaptation*, *parameter annealing*, and *sequence length extension*:

Summary token adaptation for CPT. As described earlier, summary tokens are designed to distill chunk-wise semantic information and serve as compressed priors for distant contexts. To endow these tokens with dense, informative representations, we devise a multi-granularity distillation strategy that aligns the KSA student with a vanilla full-attention teacher at three levels: *layer-wise*, *distribution-wise*, and *objective-wise*. Concretely, we instantiate the summary token $\mathcal{S}$ as a new vocabulary entry and equip each summary layer with independent attention parameters W_S^Q , W_S^K , and W_S^V .

Layer-wise attention score alignment. Let W^Q, W^K, W^V denote the pretrained attention projections, and let W_S^Q, W_S^K, W_S^V denote the inde-

pendent summary matrices introduced for KSA. Let $\mathcal{T}$ and $\hat{\mathcal{T}}$ denote the original and summary-augmented input sequences (Equation 1), $\mathcal{S} \subset \{1, \dots, |\hat{\mathcal{T}}|\}$ index the summary positions. The teacher branch projects input tensors using the pre-trained weights,

$$X = \mathcal{T} (W^X)^\top, \quad X \in \{Q, K, V\}. \quad (4)$$

The student processes its input by employing position-specific projections: text positions retain the pretrained weights, whereas summary tokens route through the newly introduced parameters,

$$\hat{X}_t = \begin{cases} \hat{\mathcal{T}}_t (W^X)^\top, & t \notin \mathcal{S}, \\ \hat{\mathcal{T}}_t (W_S^X)^\top, & t \in \mathcal{S}, \end{cases} \quad X \in \{Q, K, V\}. \quad (5)$$

Attention patterns also diverge between them. The teacher performs standard full attention:

$$O = \text{softmax}(QK^\top / \sqrt{d}) V, \quad (6)$$

whereas the student adopts the KSA pattern $\mathcal{M}_{\text{KSA}}$, in which each text token attends to its local sliding-chunk window together with preceding summary tokens, and each summary token attends only to the text chunk it compresses; as detailed in Section 2.1:

$$\hat{O} = \text{softmax}(\hat{Q}\hat{K}^\top / \sqrt{d} + \mathcal{M}_{\text{KSA}}) \hat{V}. \quad (7)$$

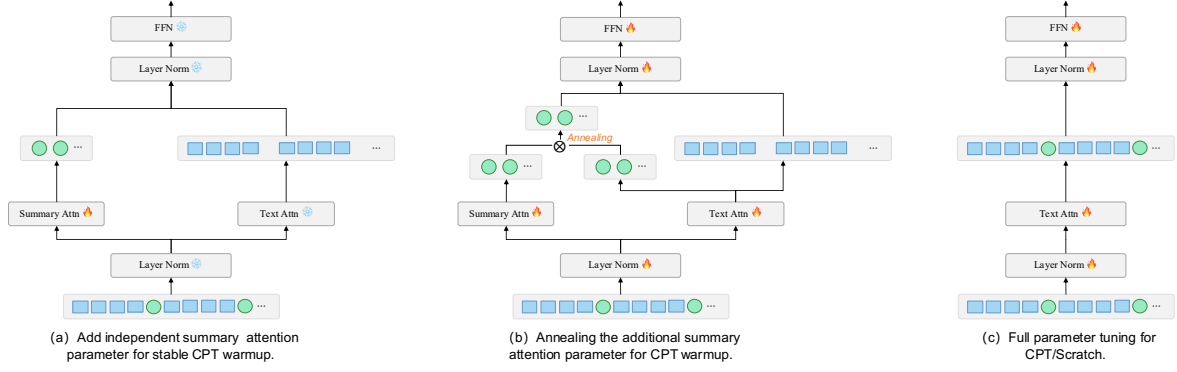


Figure 6: The training recipes for CPT warmup: (a) insert additional Q/K/V learnable weights to adapt the summary token compression; (b) conduct the annealing strategy to drop the summary weights; (c) full parameter tuning on large data corpus. Residual connections are omitted for brevity.

To align the intermediate representations, we discard the summary rows of $\hat{O}$ (which have no counterpart in $\mathcal{T}$), denoted $\hat{O}|_{\mathcal{T}} \in \mathbb{R}^{|\mathcal{T}| \times d}$, and employ a mean squared error (MSE) loss:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{L \cdot |\mathcal{T}|} \sum_{\ell=1}^L \|O_{\ell} - \hat{O}_{\ell}|_{\mathcal{T}}\|_2^2, \quad (8)$$

where L is the number of transformer blocks and $|\mathcal{T}|$ is the sequence length.

Distribution-wise regularization. While $\mathcal{L}_{\text{MSE}}$ constrains intermediate layer outputs, it does not directly enforce consistency in the final predictive distributions. We therefore introduce a KL regularizer on the output logits to provide a higher-level alignment between the KSA student and the full-attention teacher. Let W_h denote the shared LM head, and h_L and $\hat{h}_L$ be the final-layer hidden states of the teacher and student, respectively. The two predicted distributions along with KL regularizer are:

$$\begin{aligned} p &= \text{softmax}(h_L W_h^{\top}), \\ \hat{p} &= \text{softmax}(\hat{h}_L W_h^{\top}), \\ \mathcal{L}_{\text{KL}} &= \text{KL}(p \parallel \hat{p}) = \sum_v p_v \log \frac{p_v}{\hat{p}_v}. \end{aligned} \quad (9)$$

Objective-wise training. The total distillation objective combines the language modeling loss on the student with the two alignment terms to ensure warmup eventually benefits model performance:

$$\mathcal{L} = \mathcal{L}_{\text{LM}} + \alpha \mathcal{L}_{\text{MSE}} + \beta \mathcal{L}_{\text{KL}}, \quad (10)$$

where α and β are hyperparameters tuned on a validation split.

Analysis. All three loss terms, $\mathcal{L}_{\text{MSE}}$, $\mathcal{L}_{\text{KL}}$, and $\mathcal{L}_{\text{LM}}$, are computed exclusively over text token positions. This design ensures dimensional consistency between the teacher and student outputs (as the teacher lacks summary positions) while preserving semantic fidelity. Notably, restricting the loss to text positions does not cut off gradient flow to the summary parameters: *the distinctive attention pattern of text tokens, as defined in Equation (3), inherently captures the semantics of summary tokens*, enabling gradients to propagate through the attention interactions. Through these three complementary, multi-granularity distillation objectives, the independent summary parameters learn to approximate the attention distribution of vanilla full attention as closely as possible, establishing a solid foundation for subsequent full-parameter training and scaling to longer sequences.

Parameter annealing for CPT. To avoid introducing additional parameters that would increase inference cost, we propose a parameter annealing strategy that gradually absorbs the independent summary parameters into the main LLM weights. Specifically, at each summary position, we perform *two QKV projections* on the same hidden state: one using the shared LLM weights, gaining $(q_s^{\text{main}}, k_s^{\text{main}}, v_s^{\text{main}})$, and one through the independent summary weights, yielding (q_s, k_s, v_s) . The QKV triplets fed into the *single attention computation* are then linearly interpolated:

$$\tilde{x}_s = \lambda x_s + (1 - \lambda) x_s^{\text{main}}, \quad x \in \{q, k, v\}, \quad (11)$$

The interpolation coefficient follows an iteration-dependent schedule:

$$\lambda(s) = \begin{cases} 1, & s \leq s_{\text{start}}, \\ 1 - \frac{s - s_{\text{start}}}{s_{\text{end}} - s_{\text{start}}}, & s_{\text{start}} < s < s_{\text{end}}, \\ 0, & s \geq s_{\text{end}}, \end{cases} \quad (12)$$

where s denotes the current training step, and $s_{\text{start}}, s_{\text{end}}$ define the annealing window. When $s \leq s_{\text{start}}$, the summary positions are governed entirely by the independent parameters; when $s \geq s_{\text{end}}$, they rely exclusively on the main LLM weights, allowing the auxiliary parameters to be removed at inference time without any architectural modification. This strategy provides a smooth curriculum that facilitates the transition from a dedicated summary head to a fully shared representation.

Sequence length extension for CPT/Scratch.

For CPT settings, introduction of summary tokens stops at the context length of 32K, allowing the newly added parameters to learn stable representations. We then extend the context in two stages: 64K for 35B tokens, followed by 128K for 25B tokens. This staged schedule enables the summary mechanism to adapt to longer contexts incrementally, equipping KSA with the ability of handling extremely long sequences. For the Scratch setting, we directly share the attention weights of summary token and text token at the begin, and then training 250B/50B/50B/50B tokens with the sequence length 8K/32K/64K/128K. Note that we change the RoPE Theta from 10^4 to 10^6 after the 8K training finished.

B.5 Chunk-Level Sliding over Token-Level Analysis

A natural alternative is standard token-level Sliding Window Attention (SWA), where text token could attends to a fix window size latest tokens and it have been already validated in many different LLMs (e.g., GPT-OSS). However, when combined with our summary tokens, its token-level sliding may omit information or introduce more information. As shown in Figure 7, if the window boundary cuts through a chunk, the text token sees only a subset of that chunk’s text tokens. Meanwhile, the chunk is not fully outside the window, so its summary token does not qualify as a distant summary either. As a result, the partially visible chunk is neither covered by complete text nor compensated by its summary, which results in loss of information.

By sliding at chunk granularity, we guarantee a clean partition: every past chunk is either fully within the window (all text tokens visible) or fully outside (accessed only through its summary token), with no intermediate state. This information routing ensures that no chunk’s information is lost.

C More Empirical Results

C.1 RULER Subtasks

Further quantitative analysis in Table 6 reveals that **Hybrid-KSA** consistently outperforms other methods with limited effective context across multiple subtasks:

i) *KSA achieves strong NIAH performance, even compared with Full attention.* On NIAH-Multivalue, **Hybrid-KSA** attains **98.75**, a substantial gain of **+10.63** over **Full** attention (88.12) and the **highest** across all configurations. On NIAH-Multiquery, it scores 98.12, closely approaching the best result (99.38 by **Hybrid-Linear**) and surpassing **Full** attention (95.62).

ii) *Our method scales robustly to more challenging synthetic subtasks beyond simple retrieval.* **Hybrid-KSA** surpasses **+30.0** over **Full** on VT and **+14.18** over **Full** on FWE.

iii) In summary, *KSA serves as a high-fidelity compressed relay, enabling robust information retrieval and complex long-sequence reasoning across 128K tokens without the prohibitive overhead of full attention.*

C.2 Needle-in-a-Haystack

To delve deeper into KSA’s ability to retrieve specific information from long contexts, we conduct the Needle-in-a-Haystack benchmark (Martin et al., 2023). In this task, a short factual statement, *i.e.*, the *needle*, is embedded at varying depths within a long context, *i.e.*, the *haystack*; and the model is asked to retrieve it via a targeted question. The retrieval success rate across different needle positions serves as a robust indicator of long-context information retrieval capability.

Figure 8 reports single-needle retrieval accuracy across context lengths from 4K to 128K and needle depths from 0% to 100%. **Hybrid-KSA** attention maintains near-perfect retrieval across all lengths and depths, with only a slight dip at 128K, suggesting that a small dose of full attention effectively compensates for the compression loss in extremely long sequences.

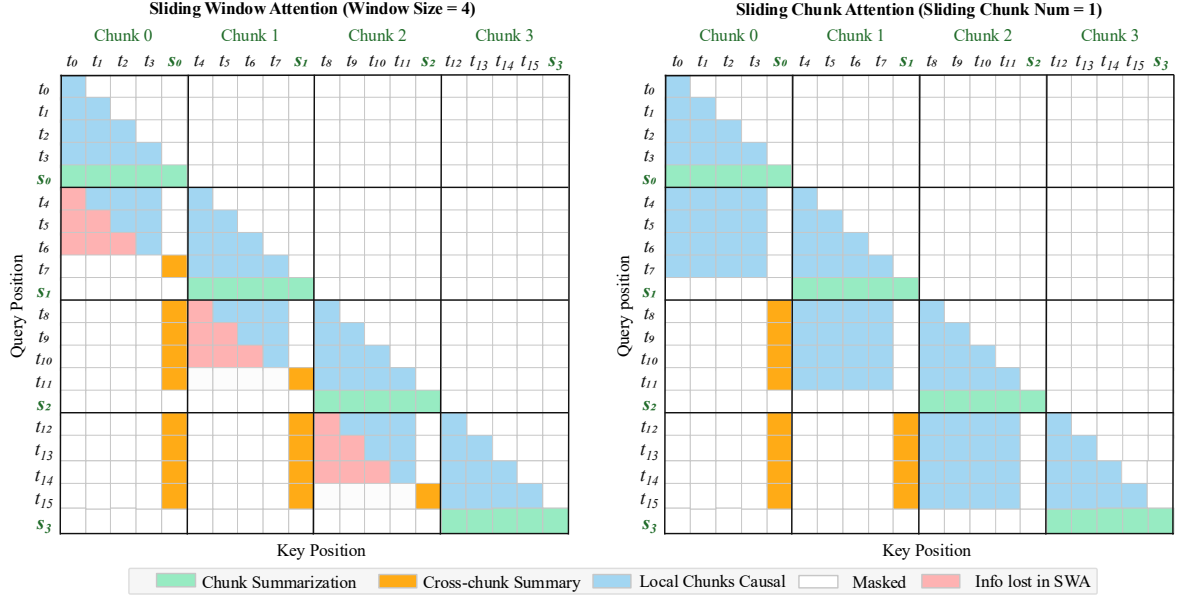


Figure 7: **Sliding Window Attention (left)** may cut through a chunk, resulting attention to overlook certain tokens at boundaries and causing information loss (red regions). **Sliding Chunk Attention (right)** aligns window boundaries with chunk boundaries, ensuring clean information routing.

Table 6: For CPT, **Hybrid-KSA** achieves the best overall results on RULER 128K subtasks, beating all other efficient baselines and **Full** attention on **NIAH-Multivalue**, **VT**, **FWE**, and **SQuAD**.

Subtasks	Baselines				Ours	
	Full	Hybrid-SWA	Hybrid-SCA	Hybrid-Linear	KSA	Hybrid-KSA
NIAH-Single	100.00	100.00	99.16	100.00	97.50	100.00
NIAH-Multikey	75.00	74.16	70.84	79.16	74.16	75.84
NIAH-Multivalue	88.12	83.75	91.25	95.62	83.75	98.75
NIAH-Multiquery	95.62	93.12	98.12	99.38	95.62	98.12
VT	60.50	67.50	42.50	87.50	65.50	90.50
FWE	51.66	51.66	33.33	23.33	72.50	65.84
SQuAD	30.00	30.00	15.00	35.00	32.50	42.50

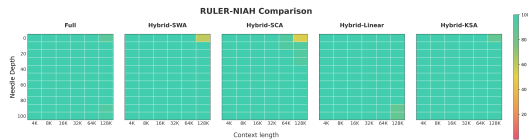


Figure 8: NIAH-Single results after converting the model into a hybrid architecture and applying continual pre-training. **Hybrid-KSA** achieves near-perfect needle retrieval up to 128K context length, with only a minor drop from full accuracy at 128K.

C.3 Training Loss and Evaluation Score Analysis

We further analyze the training dynamics of Hybrid-KSA by examining both training loss and evaluation performance across multiple benchmarks under the train-from-scratch setting.

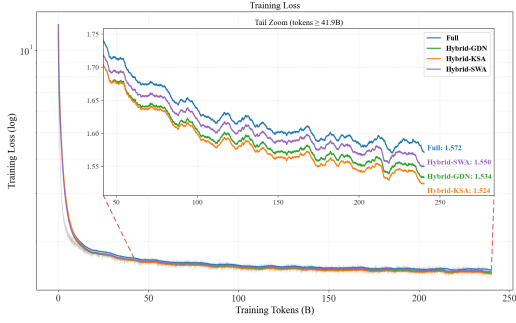
Training Loss. **Hybrid-KSA** demonstrates best converge efficiency compared to all baselines:

i) *Hybrid-KSA achieves the lowest training loss throughout training.* As shown in Figure 9a, **Hybrid-KSA** reaches a final loss of **1.524**, compared with 1.572 for **Full** attention. This difference is consistent with distinct optimization trajectories, but does not by itself establish the cause of downstream performance differences.

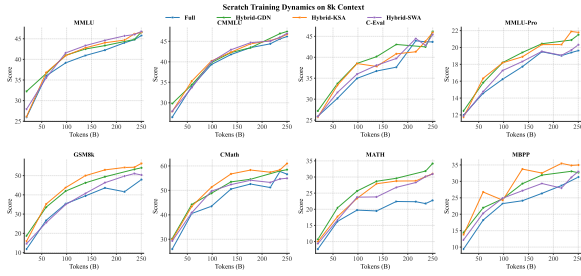
ii) *The advantage becomes more pronounced in the long-tail regime.* In the zoomed-in region (tokens $\geq 41.9B$), Hybrid-KSA consistently stays below all baselines with a stable margin.

Evaluation Scores. The improved optimization translates directly into stronger performance:

i) *Hybrid-KSA consistently matches or outperforms baselines across benchmarks.* Across knowledge (MMLU, CMMLU, C-Eval), reason-



(a) **Hybrid-KSA** has the best convergence efficiency during from-scratch training.



(b) Our method’s performance improves more rapidly than other models on multiple benchmarks.

Figure 9: **Hybrid-KSA** delivers the best training dynamics, with superior convergence efficiency and faster performance gains across benchmarks in from-scratch training.

ing (GSM8K, MATH), and code (MBPP) tasks, Hybrid-KSA demonstrates competitive or superior performance throughout training. Notably, it achieves the best final scores on several tasks, including **GSM8K**, **MATH**, and **MBPP**.

ii) *Hybrid-KSA exhibits faster and more stable performance improvement.* Compared to **Full** attention, which lags behind across most benchmarks, Hybrid-KSA shows consistently higher scores at intermediate training stages, suggesting improved learning efficiency. It also avoids the fluctuations observed in other hybrid baselines (e.g., **Hybrid-SWA**), leading to smoother and more reliable gains.

iii) *The advantage is particularly evident in reasoning-intensive tasks.* On GSM8K, CMATH, and MATH, Hybrid-KSA maintains a clear margin over **Full** attention and remains competitive with or better than other hybrid methods, indicating that the summary-based mechanism does not hinder, and may even enhance, multi-step reasoning ability.

C.4 Per-Layer Attention Pattern Analysis

To understand *why* hybrid architecture benefits long-context retrieval, we visualize the per-layer attention distributions of **KSA** and **Hybrid-KSA**

on an out-of-window NIAH example in Figure 10. We compare the first block, *i.e.*, shallow layers L0-3 and the last block *i.e.*, deep layers L24-27; within each row, the y-axis is shared across the two models so that attention magnitudes are directly comparable.

Two qualitative patterns emerge in **Hybrid-KSA** that are absent in **KSA**:

i) *Hybrid SA attends to summary tokens more frequently.* In shallow SA layers (most clearly in L2), the hybrid model develops a periodic “comb” pattern that attends to past summary tokens at every chunk boundary, while **KSA** exhibits only a single weak peak close to the needle position.

ii) *The interleaved full-attention layers (L3, L27, labeled “[F]”) serve as cross-chunk integrators.* Their attention maps show a sharp spike at the needle chunk while **KSA** remains flat, providing an explicit token-level retrieval indicator that **KSA** must instead approximate only summary tokens.

iii) *Both models focus heavily on the earliest summary tokens in the deep block with notably different shapes.* Hybrid-KSA’s sink spreads across roughly six chunks, while the pure model’s sink collapses onto only two, suggesting that the hybrid distributes its “register” role over a wider sink basin.

Taken together, *the shallow comb pattern and the full-layer integrator provide two concerted retrieval indicators that KSA lacks*, which we hypothesize is the key reason **Hybrid-KSA** retrieves out-of-window needles more reliably thus possessing robust long-context capacity.

D Related Works

In this section, we briefly review the latest efforts of long-context modeling.

D.1 Efficient Attention Variants

Prior works have explored efficient attention variants of (MHA) (Vaswani et al., 2017) from several directions. The first line of work reduces the memory footprint of KV cache. Multi-query attention (MQA) (Shazeer, 2019) lets all query heads share a single key-value head, while grouped-query attention (GQA) (Ainslie et al., 2023) forces groups of query heads to share key-value heads. Multi-head Latent Attention (MLA) (DeepSeek-V2 et al., 2024) projects KV into low-dimensional latent space, and adopts matrix absorption to store only latent vectors in cache during inference. However,

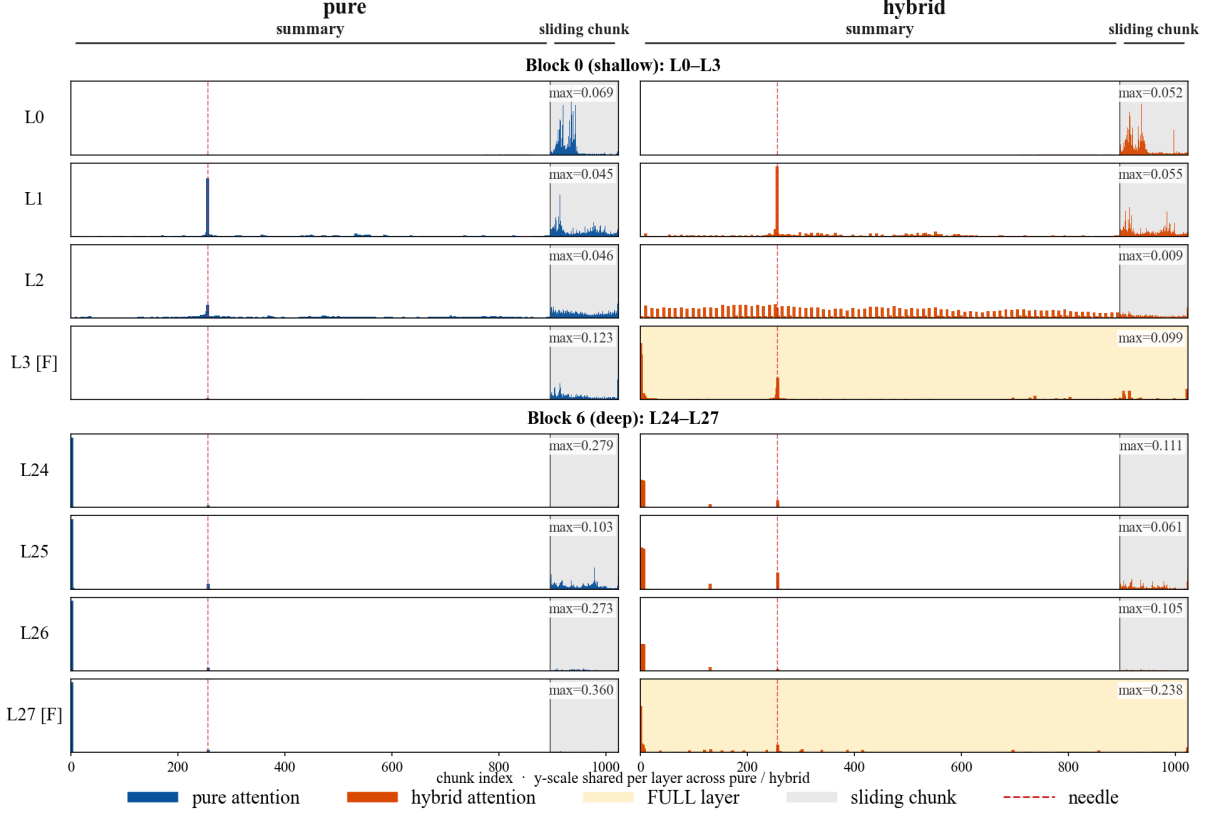


Figure 10: Per-layer attention patterns reveal that **Hybrid-KSA** enhances cross-chunk retrieval through “comb-like” summary attention and full-layer integration.

these methods still hold a quadratic complexity.

The second category reduces computation by limiting the attention scope. **Local attention** mechanisms, such as sliding-window and sliding-chunk approaches (Child et al., 2019; Beltagy et al., 2020; Jiang et al., 2023; Gemma2 et al., 2024), restrict each query to attend only to KV states within a local window of size W , reducing the complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(WN)$. **Sparse attention** variants (Yang et al., 2026; Zhang et al., 2025a; Yuan et al., 2025; Liu et al., 2025) address this limitation by selecting a small subset of K important KV tokens, and focusing only on the selected subset, thus reducing the cost to $\mathcal{O}(KN)$.

The third direction replaces full softmax attention with linear approximations. Retentive Networks (Sun et al., 2023) and Eagle (Peng et al., 2024) reformulate the softmax kernel with feature maps, avoiding explicit materialization of the full attention matrix. Mamba (Gu and Dao, 2024) and Gated Delta Net (Yang et al., 2025), model long-range dependencies through fixed-size states whose memory is not affected by context length. These methods offer attractive inference efficiency, but

compressing the entire history into a fixed-size state often leads to performance degradation.

D.2 Hybrid Attention Architecture

To balance the strengths of full attention and efficient attention, recent research has extensively explored hybrid attention architectures.

Hybrid-H3 (Fu et al., 2023) adopts SSM-based backbone with only a few attention layers. RecurrentGemma (Botev et al., 2024) combines gated linear recurrences with sliding-window attention. Mamba-2-Hybrid (Waleffe et al., 2024) constructs the model from a mixture of Mamba-2 layers, full-attention layers, and MLP layers. Jamba (Lieber et al., 2024; Team et al., 2024) further scales the hybrid paradigm by interleaving Mamba blocks with Transformer and Mixture-of-Experts (MoE) layers.

Beyond layer-level interleaving, several works introduce more structured hybridization strategies. Zamba (Glorioso et al., 2024b,a) shares a global attention block across multiple Mamba blocks. YOCO (Sun et al., 2024) separates sequence encoding to a self-decoder and generation to a cross-decoder. Hymba (Dong et al., 2025) performs finer-grained hybridization by assigning different KV

heads to softmax attention and SSM modules.

D.3 Sequence-Level Token Compression

Beyond optimizations along the layer or attention-scope dimensions, another line of work directly performs semantic-level compression along the sequence dimension by introducing a small number of learnable gist or summary tokens to condense historical contexts, so that distant information can be accessed through these compressed virtual tokens rather than full token-level history. Gisting (Mu et al., 2023) introduces learnable gist tokens to compress an instruction-following prompt into a small set of cacheable activations via a modified attention mask, enabling efficient prompt reuse across inputs. AutoCompressors (Chevalier et al., 2023) extend this idea to long-context language modeling by recursively accumulating last-layer summary vectors as soft prompts across chunks, while Activation Beacon (Zhang et al., 2025b) further pushes the compression into per-layer key-value activations and accumulates them across chunks during chunk-wise training. UniGist (Deng et al., 2025) removes the chunk-wise constraint with a unified sparse gist layout, in which every token can attend to all preceding gist tokens and a local raw window, enabling single-pass long-sequence training. More recent variants explore complementary directions on top of this line: DAST (Chen et al., 2025) dynamically reallocates a fixed soft-token budget across chunks based on information density, with the per-chunk allocation determined at inference time, while AdmTree (Li et al., 2025) organizes leaf gist tokens into an adaptive semantic tree and aggregates them with bidirectional self-attention to mitigate the unidirectional degradation of recursive compression.

Compared with the previous methods, we have the following design insights. i) Hybrid attention architecture. Instead of applying a single compressed pattern uniformly across all transformer layers, KSA interleaves summary-attention and full-attention layers at a 3:1 ratio, with the small dose of full-attention layers acting as a cross-chunk integrator (Appendix C.4). ii) Chunk-local non-recursive compression. Each summary token attends strictly to the raw tokens within its own chunk, so every raw token is compressed exactly once and no recursive summary-of-summary aggregation is involved. iii) Non-overlapping token routing. Inside the local window a text token attends only to raw tokens, while distant context is exclu-

sively accessed through summary tokens, eliminating double counting along the information path.

E KV Cache Memory Analysis

As we known, the key challenge for deploying LLM is the *KV cache state* amount bound during inference. To alleviate the cache amount issue, there are many comprehensive works were proposed, this section analyzes KSA and other methods KV cache amount in details.

E.1 Per-Token KV Cache Cost

Table 7 summarizes the KV cache state amount for each mechanism in naive setting:

Naive (MHA)	Multi-Head (Vaswani et al., 2017)	Full Attention
Full Attention stores all n past key-value pairs explicitly, resulting in a cache of size $2nhd$ that grows linearly with sequence length. Each prediction step requires attending over all n cached KV results. While this provides access to full context, the linear KV cache growth becomes the notorious bottleneck for long-sequence inference.		

Grouped Query Attention (GQA) (Ainslie et al., 2023). In standard MHA, each of the h attention heads maintains its own KV states, the GQA proposes a simple-yet-effective technique to organize the total h heads into g groups, where same group of Q-heads shared the same cached KV-heads. Thus the KV cache reduces from $2nhd$ (MHA) to $2ngd$ (GQA), the KV compression ratio is about g/h .

Multi-head Latent Attention (MLA) (DeepSeek-V2 et al., 2024). Compared with GQA, MLA takes a more aggressive approach by projecting all KV information into low-rank latent vectors. During inference: i) first introduce a nd_c cached block to store the non-RoPE key and values; the per-head non-RoPE keys and values are reconstructed via per-head up-projection matrices. ii) additionally, MLA also introduce a small nd_r cached RoPE block to decoupled for key part. With the two different cache block, the total per-token cache amount is $n(d_c + d_r)$.

Gated Delta Net (GDN) (Yang et al., 2025). Different with the MLA and GQA that follows the standard full attention mechanism, the linear GDN mechanism compresses the entire sequence history into a fixed $d \times d$ state matrix through diagonal gating. The KV cache is always a small constant value

Table 7: Next per-token KV Cache comparison. Taking the $k = 8$ chunk compression and $h = 128, d = 128, g = 8, d_c = 512, d_r = 64$ configurations as example.

Mechanism	Effective Context	KV Cache Size ($n \rightarrow \infty$)	Compression Rate
MHA	n (exactly)	$2 \cdot n \cdot h \cdot d$	-
GQA	n (exactly)	$2 \cdot n \cdot g \cdot d$	$g/h \approx 6.25\%$
MLA	n (exactly)	$n \cdot (d_c + d_r)$	$(d_c + d_r)/(2 \cdot h \cdot d) \approx 1.76\%$
GDN	n (ambiguous)	$2 \cdot h \cdot d^2$	$d/n \approx 0\%$
SWA	w (exactly)	$2 \cdot w \cdot g \cdot d$	$w/n \cdot g/h \approx 0\%$
KSA	n (summarized)	$2 \cdot n/k \cdot h \cdot d$	$1/k \approx 12.5\%$
KSA with GQA	n (summarized)	$2 \cdot n/k \cdot g \cdot d$	$1/k \cdot g/h \approx 0.78\%$
KSA with MLA (theory)	n (summarized)	$n/k \cdot (d_c + d_r)$	$1/k \cdot (d_c + d_r)/(2 \cdot h \cdot d) \approx 0.22\%$

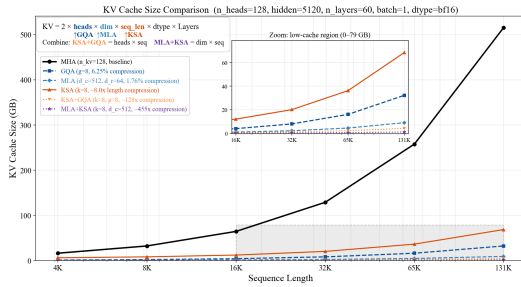


Figure 11: KV cache size comparison across different mechanisms as sequence length grows.

$2d^2$, which is independent of sequence length. Nevertheless, the long-term information dependency is unobserved and unexplainable in inference process: as new tokens arrive, older information is progressively attenuated by the decay factors α_t , making the effective context is hard to analyze.

Sliding Window Attention (SWA) (Beltagy et al., 2020). Following the standard full attention mechanism, SWA narrows the visual token window to reduce the KV cache amount. Specifically, the SWA maintains latest w recent tokens in a FIFO buffer, with a fixed cache size $2wgd$. Although effective, the limitation is also obviously: tokens relative index more than w are completely lost, could not access the long-term information directly.

Kwai Summary Attention (KSA). KSA augments a local window of $w = Ck$ raw text states with n/k summary states. Its per-layer cache is $2(w + n/k)hd$, or $2(w + n/k)gd$ with GQA. For $n \gg w$, the compression ratio approaches $1/k$, although the cache remains $\mathcal{O}(n)$ for fixed k .

E.2 Orthogonality of KV Cache Compression

Actually, the KV cache amount of **one single layer** can be represented as follows:

$$\text{KV Cache} = \text{PastToken} \times \text{HeadNum} \times \text{EmbeddingDim} \quad (13)$$

KSA acts on the sequence-state axis and is conceptually orthogonal to **GQA** head grouping and **MLA** representation compression. Our experiments combine KSA with GQA only; the compatibility and ratios for KSA with MLA are theoretical. For KSA with GQA, the compression factors multiply:

$$\text{KV Cache}_{\text{KSA+GQA}} = 2(n/k + w)gd \quad (14)$$

In theory, KSA could be combined with MLA:

$$\text{KV Cache}_{\text{KSA+MLA}} = (n/k + w)(d_c + d_r) \quad (15)$$

Table 7 reports theoretical ratios; fixed- k KSA provides constant-factor, not sub-linear, cache reduction.