NCSA: NATIVE CHUNK SUMMARY ATTENTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-context language modeling faces two bottlenecks: attention computation grows rapidly with context length, and the key–value (KV) cache grows with every token during generation. Recent KV-compression methods address these bottlenecks by representing several tokens with a compact summary. Existing designs mainly follow two approaches. Dedicated-compressor designs generate compressed KV entries through an additional module and computation path. Summary-token designs instead insert a learned summary for each chunk and process it as an ordinary sequence position. Consequently, the extra positions pass through every layer's feed-forward network, and a summary produced in one layer is not available to text attention until the next. In this work, we introduce NCSA (Native Chunk Summary Attention), a nearly parameter-free architecture that generates and uses chunk summaries within the same attention layer. NCSA keeps summaries outside the token sequence in a summary residual stream, thereby eliminating the one-layer delay and making summary-side feed-forward computation optional while preserving the compressed KV cache. At the 4K context length used for pretraining, NCSA reduces estimated Transformer-block FLOPs by 14.2% relative to the summary-token baseline. Across matched pretraining runs at 0.3B, 0.6B, and 4B parameters on 50B tokens, NCSA consistently improves long-context retrieval over the compressed-attention baselines while remaining competitive in language modeling and commonsense reasoning.

1 INTRODUCTION

Long-context modeling enables capabilities such as agentic memory, long chain-of-thought reasoning, and repository-scale code understanding, but it is expensive: full attention scales quadratically with sequence length during training and prefill, while the KV cache grows linearly during autoregressive decoding. Existing efficiency methods make different trade-offs. Sparse attention reduces attention computation but often retains raw KV states; recurrent linear attention and local windows reduce memory more aggressively but may compress or discard distant information. Sequence compression offers a middle ground by representing each token chunk with a compact summary, reducing both attention cost and KV-cache size. We focus on *native compression*, in which the compression mechanism is learned jointly with the model during pretraining.

Within this design space, native-compression methods largely follow two routes. DeepSeek-V4's HCA and CSA form content-aware memories with dedicated attention branches or per-layer compressors (DeepSeek-AI, 2026). Gist Tokens (Mu et al., 2023), Landmark Attention (Mohtashami & Jaggi, 2023), AutoCompressors (Chevalier et al., 2023), ICAE (Ge et al., 2024), and UniGist (Deng et al., 2025) instead introduce learned summary tokens or states to represent context. KSA (Chu et al., 2026) adapts this idea to native long-context compression by inserting one summary token after each chunk, avoiding a separate compressor but coupling summaries to the standard token dataflow.

The coupling has two costs. First, summary positions must pass through the token feed-forward network, regardless of whether this additional computation is beneficial for a given quality–compute target. Second, the summary KV read by remote queries in layer ℓ is projected before the summary is updated from its chunk in that layer. The newly compressed content therefore becomes visible only in layer $\ell + 1$. This *read-before-write* gap is absent from CSA, where compression precedes main text attention.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Motivated by these observations, we propose **Native Chunk Summary Attention (NCSA)**, which reorganizes how chunk summaries are constructed and used within each attention layer. NCSA maintains a summary residual stream outside the token sequence, enabling same-layer use of updated summaries and making summary-side FFN computation optional rather than mandatory. It retains $O(S/C)$ compressed memory without the one-layer delay or a dedicated compressor.

To evaluate the effectiveness of NCSA, we conduct a series of controlled experiments comparing it with dense attention, a dedicated-compressor architecture (CSA)[1], and a summary-token architecture (KSA) within the same Qwen3-based framework (Yang et al., 2025). All four models are trained with matched data and training recipes at 0.3B, 0.6B, and 4B scales, each using 50B training tokens and a 4K context length. We evaluate language modeling and commonsense reasoning for general capability, RULER (Hsieh et al., 2024) for controlled long-context performance, and real-world in-context retrieval on natural documents. Additionally, to isolate and illustrate the architectural limitation imposed by KSA's *read-before-write* ordering, we conduct a single-layer MQAR probe (Arora et al., 2023) in Appendix G. Taken together, these experiments show that NCSA improves long-context retrieval over both compressed-attention baselines at every evaluated scale while remaining competitive in language modeling and commonsense reasoning. Our contributions can be summarized as follows:

- We analyze summary-token compression through its forward and backward computation graphs, showing that parallel token updates create a *read-before-write* gap: a newly written summary is unavailable to same-layer text queries, and retrieval supervision reaches the write only through a later-layer read.
- Based on this analysis, we introduce NCSA, an attention architecture that writes and reads chunk summaries within the same layer, without a dedicated compressor or inserted summary positions.
- Through matched comparisons across three model scales, we show that NCSA consistently improves long-context retrieval over compressed-attention baselines while preserving general language capability and reducing the computation associated with summary-token processing.

2 PRELIMINARIES

2.1 NATIVE COMPRESSION ATTENTION

Let $X^\ell = [x_1^\ell, \ldots, x_S^\ell] \in \mathbb{R}^{S\times d}$ denote the text states entering Transformer (Vaswani et al., 2017) layer ℓ. We partition the sequence into $G = \lceil S/C \rceil$ chunks, where $\mathcal{C}_j^\ell = X^\ell_{(j-1)C+1:\min(jC,S)}$. For any text or summary state h, define

$$\bar{h}^\ell = \mathrm{Norm}_1^\ell(h), \qquad q^\ell(h) = \bar{h}^\ell W_Q^\ell, \quad k^\ell(h) = \bar{h}^\ell W_K^\ell, \quad v^\ell(h) = \bar{h}^\ell W_V^\ell. \tag{1}$$

We use the same notation when these projections are applied row-wise to a matrix. KSA and NCSA use the layer's existing attention projections for both text and summary states; no separate summary projection is introduced. Their chunk memories and the mechanism that constructs them are learned jointly under the language-modeling objective. A text query reads recent tokens in their original form and distant chunks through compressed key–value entries:

$$\mathcal{K}_i^\ell = [K^\ell_{\mathcal{L}_i}; K^\ell_{\mathcal{R}_i}], \qquad \mathcal{V}_i^\ell = [V^\ell_{\mathcal{L}_i}; V^\ell_{\mathcal{R}_i}]. \tag{2}$$

Here, i denotes the query position, $\mathcal{L}_i$ indexes the causal text positions retained in raw form for that query, and $\mathcal{R}_i$ indexes earlier chunks available through compressed KV. With a fixed local window W, this organization retains $O(W + S/C)$ entries per layer while keeping distant chunks individually addressable. Architectures in this family differ mainly in how they produce and update the compressed representations.

[1]Our CSA baseline retains the per-layer compressor but omits DeepSeek-V4's indexer and top-k sparse selection, attending instead to all compressed entries.

2.2 SUMMARY-TOKEN COMPRESSION

Gist-based methods compress context by augmenting the token sequence with learnable summary tokens. KSA adopts this design during pretraining, placing one summary token z_j^ℓ after each chunk $\mathcal{C}_j^\ell$ of C tokens. Assuming C divides S for clarity, its layer-ℓ input is

$$\widetilde{X}^\ell = [\mathcal{C}_1^\ell, z_1^\ell, \ldots, \mathcal{C}_G^\ell, z_G^\ell], \qquad G = S/C, \quad \widetilde{X}^\ell \in \mathbb{R}^{(S+G)\times d}. \tag{3}$$

All summary tokens are initialized from the same learned embedding but develop chunk-specific hidden states after attending to their respective chunks. KSA then sends text and summary positions through the same pre-normalized Transformer layer. Let H^ℓ denote the joint post-attention state. Using Attn_{M^ℓ} for scaled dot-product attention under KSA's mask M^ℓ, the shared layer update is

$$H^\ell = \widetilde{X}^\ell + \mathrm{Attn}_{M^\ell}(Q^\ell, K^\ell, V^\ell)W_O^\ell, \qquad \widetilde{X}^{\ell+1} = H^\ell + \mathrm{FFN}^\ell\Big(\mathrm{Norm}_2^\ell(H^\ell)\Big). \tag{4}$$

Equation 4 shows that text and summary rows share the same Transformer dataflow: both are updated by masked attention and then by the feed-forward network. We use H_x^ℓ and H_z^ℓ for the text and summary rows of H^ℓ, respectively. Their only distinction within attention is the visibility pattern in Figure 1: summaries aggregate their assigned chunks, whereas text positions combine recent raw context with compressed memory from earlier chunks. The figure uses a one-chunk raw window for readability; the main experiments use 128 raw tokens.

This shared update rule determines when a summary can incorporate its chunk and when the resulting content becomes available to other positions. The next section formalizes this ordering in both the forward and backward computation graphs.

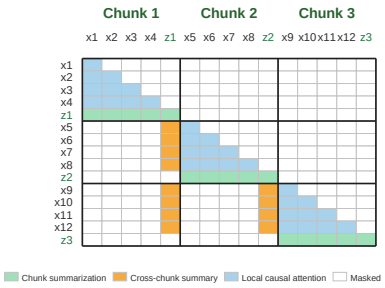


Figure 1: **Full KSA attention mask.** Here $C = 4$ with one raw chunk.

3 ARCHITECTURAL ANALYSIS OF SUMMARY-TOKEN COMPRESSION

3.1 READ-BEFORE-WRITE ORDERING IN SUMMARY-TOKEN ATTENTION

A Transformer layer projects every input position into queries, keys, and values before producing any attention output. We examine how this parallel update orders a summary's aggregation of its chunk relative to its use by distant text queries; we refer to the resulting execution order as *read-before-write*.

Collect the incoming summaries as $Z^\ell = [z_1^\ell, \ldots, z_G^\ell] \in \mathbb{R}^{G\times d}$. For text position i and chunk j, let $q_i^\ell = q^\ell(x_i^\ell)$ and $q_j^\ell = q^\ell(z_j^\ell)$ denote the text and summary queries, respectively. We further denote the incoming summary KV by $(k_j^\ell, v_j^\ell) = (k^\ell(z_j^\ell), v^\ell(z_j^\ell))$ and the chunk KV by $(K_{\mathcal{C}_j}^\ell, V_{\mathcal{C}_j}^\ell) = (k^\ell(\mathcal{C}_j^\ell), v^\ell(\mathcal{C}_j^\ell))$. The summary-write and text-read paths are

$$\begin{aligned}
&(K_{\mathcal{R}_i}^\ell, V_{\mathcal{R}_i}^\ell) = \big(k^\ell(Z_{\mathcal{R}_i}^\ell), v^\ell(Z_{\mathcal{R}_i}^\ell)\big), && (K_{\mathcal{L}_i}^\ell, V_{\mathcal{L}_i}^\ell) = \big(k^\ell(X_{\mathcal{L}_i}^\ell), v^\ell(X_{\mathcal{L}_i}^\ell)\big),\\
&\underbrace{H_z^\ell[j] = z_j^\ell + \mathrm{Attn}\Big(q_j^\ell, [K_{\mathcal{C}_j}^\ell; k_j^\ell], [V_{\mathcal{C}_j}^\ell; v_j^\ell]\Big) W_O^\ell}_{\text{summary write}} && \underbrace{H_x^\ell[i] = x_i^\ell + \mathrm{Attn}\big(q_i^\ell, \mathcal{K}_i^\ell, \mathcal{V}_i^\ell\big) W_O^\ell}_{\text{remote read}},\\
&\widetilde{X}^{\ell+1} = H^\ell + \mathrm{FFN}^\ell\Big(\mathrm{Norm}_2^\ell(H^\ell)\Big).
\end{aligned} \tag{5}$$

Equation 5 makes this ordering explicit. Local KV is projected from the current text states X^ℓ, while remote KV is projected from the summaries Z^ℓ entering the layer. Because summary writing and remote reading are computed in parallel, $H_z^\ell[j]$ cannot affect $H_x^\ell[i]$ until layer $\ell + 1$. The constraint is clearest at layer zero, where every z_j^0 is the same learned initialization and contains no chunk-specific information. Appendix G isolates this setting with a single-layer MQAR probe, showing that KSA cannot retrieve information written to a summary within the same layer.

Forward observation. In every layer, summary-token compression requires the shared attention projections to jointly process current text states and compressed states whose chunk content was formed one layer earlier.

3.2 Gradient consequences of delayed summary use

We next trace how this ordering affects credit assignment from the token-level loss L_i^{loss} at text position i. For a visible summary $j \in \mathcal{R}_i$, let $\mathbf{m}_j^\ell = (k_j^\ell, v_j^\ell)$ denote its incoming KV pair. Let $\Theta^\ell = \{W_Q^\ell, W_K^\ell, W_V^\ell, W_O^\ell\}$, $\Theta_{KV}^\ell = (W_K^\ell, W_V^\ell)$, and $\mathbf{J}_{a \leftarrow b} = \partial \operatorname{vec}(a)/\partial \operatorname{vec}(b)$. We denote the gradient contribution through the incoming-summary read as $g_{\text{read}}^\ell = [\nabla_{\Theta_{KV}^\ell} L_i^{\text{loss}}]_{\mathbf{m}_j^\ell \to H_x^\ell[i]}$, and the contribution through a summary written in layer ℓ and read in layer t as $g_{\text{write}}^{\ell \to t} = [\nabla_{\Theta^\ell} L_i^{\text{loss}}]_{H_z^\ell[j] \to H_x^t[i]}$.The three relevant paths in the backward graph are

$$
\begin{aligned}
g_{\text{read}}^\ell &= \mathbf{J}_{L_i^{\text{loss}} \leftarrow H_x^\ell[i]} \mathbf{J}_{H_x^\ell[i] \leftarrow \mathbf{m}_j^\ell} \mathbf{J}_{\mathbf{m}_j^\ell \leftarrow \Theta_{KV}^\ell} \not\equiv \mathbf{0}, \\
g_{\text{write}}^{\ell \to \ell} &= \mathbf{J}_{L_i^{\text{loss}} \leftarrow H_x^\ell[i]} \underbrace{\mathbf{J}_{H_x^\ell[i] \leftarrow H_z^\ell[j]}}_{\mathbf{0}} \mathbf{J}_{H_z^\ell[j] \leftarrow \Theta^\ell} = \mathbf{0}, \\
g_{\text{write}}^{\ell \to \ell+1} &= \mathbf{J}_{L_i^{\text{loss}} \leftarrow H_x^{\ell+1}[i]} \mathbf{J}_{H_x^{\ell+1}[i] \leftarrow \mathbf{m}_j^{\ell+1}} \\
&\quad \cdot \mathbf{J}_{\mathbf{m}_j^{\ell+1} \leftarrow z_j^{\ell+1}} \mathbf{J}_{z_j^{\ell+1} \leftarrow H_z^\ell[j]} \mathbf{J}_{H_z^\ell[j] \leftarrow \Theta^\ell} \not\equiv \mathbf{0}.
\end{aligned}
\tag{6}
$$

The first line is the direct gradient through the incoming summary memory. The second line vanishes because $H_x^\ell[i]$ does not depend on the concurrently written $H_z^\ell[j]$. The final path is the earliest nonzero route from that write to a remote read: $H_z^\ell[j]$ must become the next-layer summary $z_j^{\ell+1}$, be projected into $\mathbf{m}_j^{\ell+1}$, and then be read by $H_x^{\ell+1}[i]$.

Backward observation. The current-layer text loss has no direct computational path to the summary state written concurrently. Retrieval supervision reaches that state only after it is carried forward, re-projected, and read in a later layer. Shared attention parameters may still receive gradients through other paths, but those paths do not supervise the content of the current summary write through a same-layer read. This delayed credit assignment motivates a write-before-read design in which the current summary update participates in the same layer's text attention.

4 Native Chunk Summary Attention

4.1 Overview

NCSA implements the write-before-read ordering suggested by Section 3. It preserves one addressable memory per chunk in a summary residual stream outside the token sequence. As illustrated in Figure 2, text follows the standard Transformer residual and feed-forward path, whereas the separate summary stream persists across layers without extending the sequence and makes summary-side feed-forward processing optional. Within each layer, NCSA first updates each summary by attending to its corresponding chunk and then makes the updated summary KV available to text queries in the same layer.

4.2 Same-layer summary writing and reading

In NCSA, all summaries are initialized from a single learned anchor $a \in \mathbb{R}^d$, such that $z_j^0 = a$ for every chunk j. Using the projections and local/remote partition defined above, NCSA executes the following operations.

$$
\begin{aligned}
\widetilde{z}_j^\ell &= z_j^\ell + \operatorname{Attn}\Big(q^\ell(z_j^\ell), K_{\mathcal{C}_j}^\ell, V_{\mathcal{C}_j}^\ell\Big) W_O^\ell, \qquad \widetilde{\mathbf{m}}_j^\ell = \big(k^\ell(\widetilde{z}_j^\ell), v^\ell(\widetilde{z}_j^\ell)\big), \\
(K_{\mathcal{R}_i}^\ell, V_{\mathcal{R}_i}^\ell) &= \operatorname*{stack}_{j \in \mathcal{R}_i} \widetilde{\mathbf{m}}_j^\ell, \qquad H_x^\ell[i] = x_i^\ell + \operatorname{Attn}\big(q_i^\ell, \mathcal{K}_i^\ell, \mathcal{V}_i^\ell\big) W_O^\ell, \\
x_i^{\ell+1} &= H_x^\ell[i] + \operatorname{FFN}^\ell\Big(\operatorname{Norm}_2^\ell(H_x^\ell[i])\Big), \qquad z_j^{\ell+1} = \widetilde{z}_j^\ell.
\end{aligned}
\tag{7}
$$

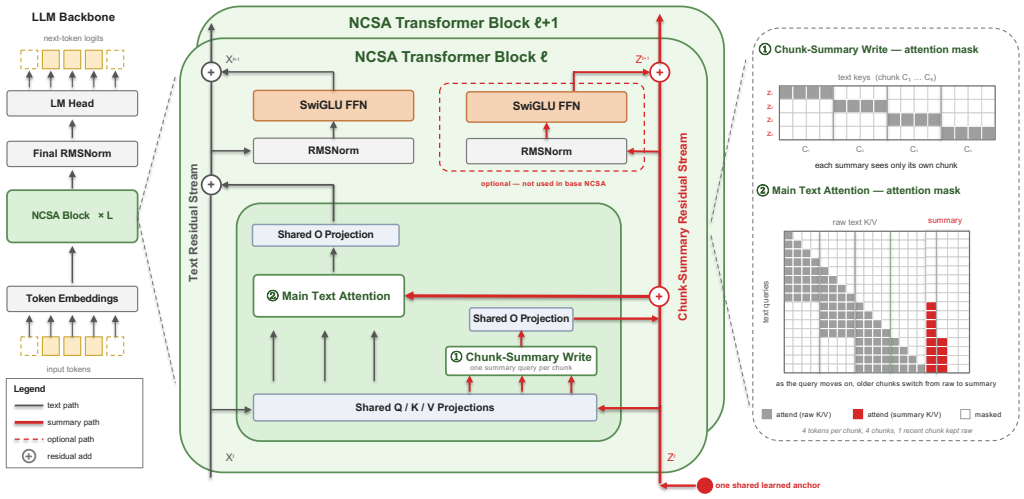


Figure 2: **NCSA architecture.** The summary residual stream remains outside the token sequence; base NCSA bypasses the feed-forward network, while the dashed path denotes the optional summary-side FFN. A single learned anchor initializes all runtime summary states.

Unlike the parallel operations in Eq. 5, Eq. 7 establishes the same-layer dependency $\widetilde{z}_j^\ell \to \widetilde{\mathbf{m}}_j^\ell \to H_x^\ell[i]$: the updated summary enters the remote KV block before main text attention. The final line makes the two residual paths explicit: text undergoes the standard FFN update, whereas the summary bypasses the FFN. Algorithm 1 provides the complete layerwise execution order.

For a summary j visible to query i, the same-layer dependency in Eq. 7 yields the gradient path

$$\underbrace{\left[\nabla_{\Theta^\ell} L_i^{\text{loss}}\right]_{\widetilde{z}_j^\ell \to H_x^\ell[i]} = \mathbf{J}_{L_i^{\text{loss}} \leftarrow H_x^\ell[i]} \mathbf{J}_{H_x^\ell[i] \leftarrow \widetilde{\mathbf{m}}_j^\ell} \mathbf{J}_{\widetilde{\mathbf{m}}_j^\ell \leftarrow \widetilde{z}_j^\ell} \mathbf{J}_{\widetilde{z}_j^\ell \leftarrow \Theta^\ell} \not\equiv \mathbf{0}}_{\text{same-layer supervision of the summary write}}. \tag{8}$$

Here W_Q^ℓ forms the summary query, W_K^ℓ, W_V^ℓ form the chunk KV and project the updated summary into memory, and W_O^ℓ maps the summary-attention output back to the residual stream. A remote-retrieval signal can therefore supervise both how the current summary is formed and how its updated state is exposed as KV within the same layer.

4.3 Attention Mask and KV-Cache Organization

The two mask panels in Figure 2 visualize NCSA's two attention stages. During chunk-summary writing, each summary query attends exactly to the C text tokens in its own chunk and, unlike KSA, not to itself. During main text attention, each query reads recent tokens in raw form and earlier chunks through summary KV. These local and remote sets follow the disjoint partition in Eq. 2, so no chunk appears in both forms. During prefill, all completed chunks are summarized in parallel. During autoregressive decoding, a new summary is formed when a chunk closes and is then appended to the per-layer KV cache. The cache therefore retains the W most recent token entries together with one entry per completed distant chunk, for a total size of $O(W + S/C)$ per layer.

4.4 Parameter Overhead and FFN Savings

The base NCSA configuration introduces only the shared d-dimensional anchor a and no dedicated compressor. Let N_{layer} be the number of Transformer layers and f the intermediate width of the Qwen3 SwiGLU FFN. Its dominant matrix-multiplication cost at one position is $F_{\text{FFN}} = 6df$ when a multiply–add counts as two FLOPs. Relative to summary-token attention with the same chunk size and visibility pattern, NCSA removes this FFN computation from one summary position per chunk:

$$\Delta F_{\text{token}} \simeq \frac{N_{\text{layer}}}{C} F_{\text{FFN}}, \qquad \Delta F_{\text{sequence}} \simeq S\, \Delta F_{\text{token}}. \tag{9}$$

Up to the negligible removal of summary self-attention, the saving per text token is independent of sequence length, while the total saving grows linearly with S. At the 4K pretraining length, our 4B configuration requires an estimated 8.10 rather than 9.44 Transformer-block forward GFLOPs per text token, a 14.2% reduction relative to KSA. These figures count matrix multiplications within the Transformer blocks; the final vocabulary projection and lower-order element-wise operations are omitted.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Training setup. We conduct controlled language-model pretraining experiments with decoder-only Transformers built in a common Qwen3-based framework. Our main comparison spans 0.3B, 0.6B, and 4B model scales. Every model is randomly initialized and trained for 50B tokens from the 150B-token sample of the Dolma 3 mixture released by the Allen Institute for AI (AI2) as part of the OLMo 3 project (Ettinger et al., 2025)[2] using a context length of 4096. At each scale, all four architectures use the same backbone, tokenizer, data order, optimization recipe, and number of updates; models compared at a given step have therefore processed exactly the same minibatches. We use Adam (Kingma & Ba, 2015) with a constant learning rate of 3×10^{-4} after 2,000 warmup steps and a global batch of 512. Full configurations are provided in Appendix B.

Baselines. We compare NCSA with standard dense causal attention, CSA with a dedicated KV compressor, and KSA with summary tokens processed along the token sequence. CSA, KSA, and NCSA all compress every four-token chunk into one KV entry and apply compression in every Transformer layer. Backbone dimensions, training data, and context length are otherwise held fixed. The comparison therefore isolates how the three compressed architectures construct, update, and expose their long-range representations.

Evaluation. We evaluate language-modeling quality with WikiText (Merity et al., 2017) and LAMBADA (Paperno et al., 2016) perplexity, general capabilities on six zero-shot commonsense tasks, controlled long-context capabilities with RULER, and six recall-intensive in-context learning tasks following the JRT evaluation setup (Arora et al., 2024). We compare all four architectures throughout these evaluations and in the position-wise analysis. Detailed protocols are provided in Appendix B.4.

5.2 MAIN RESULTS

Language modeling and common-sense reasoning. We first evaluate whether native compression preserves general language capability. We report perplexity on WikiText (Merity et al., 2017) and LAMBADA (Paperno et al., 2016), LAMBADA accuracy, and zero-shot accuracy on HellaSwag (Zellers et al., 2019), ARC-Easy and ARC-Challenge (Clark et al., 2018), PIQA (Bisk et al., 2020), WinoGrande (Sakaguchi et al., 2020), and RACE (Lai et al., 2017).

Table 1 shows consistent gains over KSA: NCSA improves the commonsense average and both LAMBADA metrics at all three scales, while also reducing WikiText perplexity at 0.6B and 4B. It achieves the strongest commonsense average overall at 0.6B and 4B. Together with the matched-data training curves in Appendix C, this pattern is consistent with the optimization argument in Section 3: making each newly written summary available to same-layer text queries provides more direct supervision for learning compressed representations, without sacrificing general language-modeling quality.

Long-context capabilities. Compressed attention creates an information bottleneck: each long-range entry must preserve the details needed by later queries. We therefore use RULER, a controlled benchmark spanning retrieval, variable tracing, aggregation, and question answering, to compare how effectively the four attention architectures preserve and use long-range information under the same training setting.

[2] https://huggingface.co/datasets/allenai/dolma3_mix-150B-1025

Table 1: Language-modeling and zero-shot commonsense results. Avg. is the mean of the six commonsense tasks. Best and second-best results within each scale are shown in **bold** and underlined.

Scale	Architecture	Params	Commonsense reasoning ↑							Language modeling		
			Hella.	ARC-E	ARC-C	PIQA	Wino.	RACE	Avg.	Wiki PPL ↓	LAMB. PPL ↓	LAMB. Acc. ↑
0.3B	Dense	341.2M	36.67	50.42	**27.30**	64.96	50.67	29.67	43.28	**26.191**	25.551	37.75
	CSA	374.8M	**37.72**	**51.77**	26.79	**65.83**	50.67	28.80	**43.60**	27.790	**21.600**	**39.96**
	KSA	341.2M	37.16	49.54	24.83	65.56	50.12	**30.24**	42.91	26.750	26.470	37.01
	NCSA (ours)	341.2M	37.14	50.25	25.43	64.53	**52.33**	29.47	43.19	26.817	26.247	37.20
0.6B	Dense	596.0M	43.07	55.26	**29.86**	67.30	53.43	32.25	46.86	**21.6178**	16.6942	43.39
	CSA	713.7M	**43.79**	56.61	29.61	**68.17**	52.49	31.96	47.10	22.3600	**13.2000**	**46.38**
	KSA	596.1M	42.72	55.05	28.92	67.68	52.64	31.67	46.45	22.1019	17.2604	44.05
	NCSA (ours)	596.1M	43.37	**57.20**	28.84	67.57	**54.38**	**32.73**	**47.35**	21.9397	15.3979	45.70
4B	Dense	4.022B	56.95	66.04	36.95	72.25	**58.72**	**36.17**	54.51	**15.3900**	7.1700	57.91
	CSA	4.400B	**57.10**	66.12	37.20	72.52	58.64	33.59	54.20	15.8064	**6.5690**	**59.60**
	KSA	4.022B	55.78	66.88	36.77	70.51	57.77	35.22	53.82	15.9133	7.8070	56.82
	NCSA (ours)	4.022B	56.88	**68.43**	**38.74**	**72.85**	58.64	35.41	**55.16**	15.6607	7.6436	56.98

Table 2: RULER results at 4K (%). Retrieval averages eight needle-in-a-haystack (NIAH) tasks; Tracing is measured by Variable Tracking (VT); Aggregation averages Common Words Extraction (CWE) and Frequent Words Extraction (FWE); QA averages SQuAD and HotpotQA; and Avg13 averages all 13 tasks.

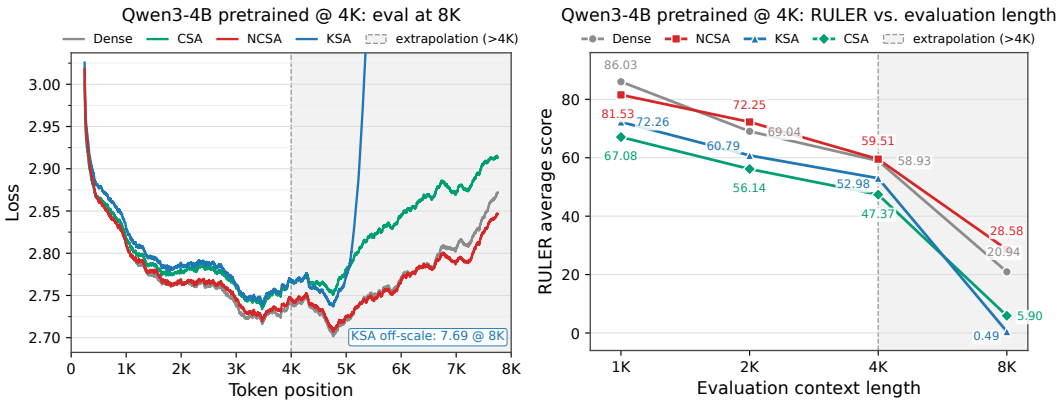
Scale	Architecture	Retrieval (8 tasks)	Tracing (VT)	Aggregation (2 tasks)	QA (2 tasks)	Avg13
0.3B	Dense	**52.49**	1.40	11.94	**25.34**	**38.14**
	CSA	34.08	0.28	0.08	13.77	23.12
	KSA	44.00	6.60	**13.01**	19.16	32.53
	NCSA (ours)	46.65	**10.76**	12.05	21.40	34.68
0.6B	Dense	**61.99**	6.16	**34.76**	**32.61**	**48.98**
	CSA	41.99	1.64	29.88	21.65	33.89
	KSA	48.87	1.08	31.12	26.89	39.08
	NCSA (ours)	54.65	**6.84**	31.17	27.56	43.19
4B	Dense	**72.34**	16.88	45.39	**39.89**	58.93
	CSA	57.61	9.88	37.56	34.95	47.37
	KSA	70.02	2.28	26.18	36.94	52.98
	NCSA (ours)	70.49	**22.12**	**54.36**	39.41	**59.51**

Table 2 shows a consistent scaling advantage for NCSA among the compressed architectures. Its Avg13 exceeds KSA by 2.15, 4.11, and 6.53 points at 0.3B, 0.6B, and 4B, respectively, with the margin widening as model size increases. At 4B, NCSA outperforms KSA in every capability group and essentially matches Dense. NCSA therefore provides the strongest long-context quality among the architectures with the same compression setting and translates increased model scale into larger gains over summary-token compression. Appendix D reports all 13 tasks.

Real-world in-context retrieval. In addition to controlled synthetic tasks, we evaluate the four architectures on six recall-intensive real-world tasks following the Just Read Twice (JRT) evaluation setup (Arora et al., 2024). At 2K, we evaluate SWDE, FDA, SQuAD, TriviaQA, Natural Questions, and DROP. Table 3 shows that NCSA is the strongest compressed architecture at every scale, improving over KSA by

Table 3: Average JRT score at 2K (%).

Architecture	0.3B	0.6B	4B
Dense	**38.17**	**43.80**	**51.62**
CSA	28.36	39.09	46.13
KSA	31.86	40.01	49.21
NCSA	34.73	40.78	49.60



(a) Position-wise PG19 loss.

(b) RULER across context lengths.

Figure 3: **Length generalization at 4B.** All four architectures were pretrained at 4K. **(a)** Position-wise loss on 8K PG19 windows, averaged over 300 windows with a 501-position running mean. **(b)** Average RULER score across evaluation lengths.

2.87, 0.77, and 0.39 points. Dense remains strongest overall, while base NCSA preserves more real-document retrieval ability than the other compressed baselines without summary-side FFN computation. Appendix E provides per-task results and protocol details.

Per-position loss and length generalization. Following Lin et al. (2025), we use position-wise teacher-forced loss to examine whether models continue to benefit from progressively longer histories. A decreasing loss profile indicates that additional preceding context remains useful, whereas a plateau or increase suggests that the model no longer exploits increasingly distant context effectively. We average loss over 300 non-overlapping PG19 (Rae et al., 2020) windows and apply a 501-position running mean for visualization. We pair this token-level measure of context utilization with RULER scores across context lengths, which provide a complementary task-level measure of length generalization; Figure 3 presents both views side by side.

Within 4K, all architectures exhibit similar position-wise profiles. Beyond the training length, NCSA remains stable and closely tracks Dense, CSA rises gradually, and KSA diverges at around 5K. RULER shows the same task-level pattern: NCSA reaches 28.58 at 8K, compared with 20.94 for Dense and 0.49 for KSA. Together, these results indicate that NCSA continues to exploit distant context more reliably. Complete position-wise results are reported in Appendix H.

5.3 Ablation Study

We use 0.6B ablations to validate the architectural choices of NCSA. The component ablations examine the summary stream, its residual update, and the optional summary-side FFN. The 3:1 hybrid instead moves beyond a homogeneous NCSA stack: three NCSA layers retain compressed memory access, while every fourth full-attention layer restores direct token-to-token interaction. This setting tests whether NCSA can serve as the dominant layer type in a practical heterogeneous backbone.

Findings. Table 4 supports the default NCSA design and shows that it remains effective in a heterogeneous attention stack. Complete per-task results are reported in Appendix F.

Summary-stream architecture. Base NCSA carries a residual summary stream across layers. *No stream* reinitializes summaries at every layer, while *No residual* retains cross-layer propagation but removes residual accumulation. Both variants broadly underperform base NCSA, supporting the persistent stream and its residual update.

Effect of the summary-side FFN. KSA summaries necessarily follow the token FFN path, whereas NCSA makes this computation optional by separating the summary stream from the token sequence (Figure 2). *Summary FFN* applies the shared FFN after forming the current summary KV and therefore preserves same-layer write–read ordering. It raises RULER Avg13 by 1.96 points in this

configuration, but slightly reduces general capability and adds one FFN evaluation per C tokens. We therefore retain the FFN-free variant as the efficiency-oriented base configuration.

Heterogeneous attention. Efficient backbones increasingly combine alternative mechanisms with full attention; Qwen3-Next, for example, uses a 3:1 Gated DeltaNet–full-attention layout (Qwen, 2025). Our *3:1 hybrid* similarly repeats three NCSA layers and one full-attention layer, retaining compressed processing while periodically restoring direct token-level communication. Its general-capability average and overall RULER score closely match Dense, showing that NCSA integrates effectively into this heterogeneous layout.

Table 4: Architecture ablations at 0.6B after 50B training tokens. Avg6 is the mean of six zero-shot commonsense tasks; RULER metrics are reported at 4K (%).

Variant	Avg6 ↑	Wiki PPL ↓	LAMB. PPL ↓	Retrieval	VT	Aggreg.	QA	Avg13
Dense	<u>46.86</u>	<u>21.62</u>	16.69	**61.99**	6.16	34.76	**32.61**	**48.98**
NCSA	**47.35**	21.94	<u>15.40</u>	54.65	6.84	31.17	27.56	43.19
No stream	46.29	22.90	18.00	52.11	5.88	36.32	27.79	42.38
No residual	46.57	22.00	**14.96**	50.53	<u>10.12</u>	32.55	<u>29.59</u>	41.43
Summary FFN	46.04	21.98	15.67	55.89	**13.24**	<u>36.41</u>	26.93	45.15
3:1 hybrid	46.51	**21.47**	15.81	<u>60.50</u>	0.68	**45.16**	29.02	<u>48.70</u>

6 Related Work

Inference-time KV-cache compression. Inference-time methods modify the cache policy of pretrained models: StreamingLLM (Xiao et al., 2024) retains attention sinks and recent tokens; H_2O (Zhang et al., 2023), Scissorhands (Liu et al., 2023), SnapKV (Li et al., 2024), and PyramidKV (Cai et al., 2024) select salient entries; KIVI (Liu et al., 2024b) quantizes KV states; and MiniCache (Liu et al., 2024a) merges representations across layers. These methods are complementary to architectures that learn compressed memory during pretraining.

A related but distinct line learns bottleneck states during training. Gist Tokens (Mu et al., 2023) add trainable gist tokens that encode a prompt span for later reuse, while Landmark Attention (Mohtashami & Jaggi, 2023) assigns learned landmark positions to context blocks to support compressed access over long inputs. AutoCompressors (Chevalier et al., 2023) recursively pass compressed summaries between context segments, and ICAE (Ge et al., 2024) uses latent memory tokens in an encoder–decoder formulation to represent a compressed context. UniGist (Deng et al., 2025) generalizes the gist-token interface across context-compression settings with hardware-aware sequence layouts.

Native attention architectures. Another line integrates compression directly into the attention dataflow. NSA (Yuan et al., 2025) combines compressed, selected, and sliding-window attention. DeepSeek-V4's HCA and CSA (DeepSeek-AI, 2026) add dedicated per-layer compression paths with different attention patterns. KSA (Chu et al., 2026) instead places one learned summary token after each chunk, so chunk memories remain explicit sequence positions and follow the standard attention and feed-forward dataflow.

7 Conclusion

We introduced NCSA, which maintains one summary per chunk in a residual stream outside the token sequence and exposes its newly projected KV to text attention within the same layer. This design preserves explicit compressed long-range memory, avoids inserted summary positions and the read-before-write gap caused by parallel token updates, and makes summary-side FFN computation optional. Matched comparisons across scales, long-context benchmarks, real-world retrieval, and position-wise analysis show the effectiveness and scalability of this attention-native compression design.

AI USE STATEMENT

The authors used generative AI tools solely to polish the language of the manuscript. The tools were not used for research ideation or execution, literature retrieval or discovery, drafting manuscript sections, generating synthetic datasets, or proving mathematical claims. All AI-assisted edits were reviewed by the authors, who verified the technical content, claims, citations, and reported results and take full responsibility for the final content of this work.

ETHICS STATEMENT

This work studies the efficiency of long-context language models and does not introduce a new dataset, collect personal information, or involve human participants. All experiments use publicly available text corpora and benchmarks under their established terms of use. Like other improvements to language-model efficiency, NCSA may reduce the cost of deploying capable models and therefore carries the broader risks associated with such systems, including misuse, harmful generation, and the reproduction of biases present in training data. The proposed architecture does not directly address these risks; models using it should retain appropriate data governance, evaluation, access controls, and deployment safeguards.

REPRODUCIBILITY STATEMENT

The NCSA architecture, attention mask, and computational-cost derivation are specified in Section 4, with forward-pass pseudocode in Appendix A. Appendix B documents the model and compression configurations, data preparation, optimization hyperparameters, random seed, hardware, checkpoint selection, and evaluation protocols. Complete task-level results and the protocols for the diagnostic analyses are provided in the remaining appendices. The controlled comparisons use matched backbones, tokenized examples in the same order, training budgets, and optimization settings across architectures.

REFERENCES

Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Zou, Atri Rudra, and Christopher Ré. Zoology: Measuring and improving recall in efficient language models. *arXiv preprint arXiv:2312.04927*, 2023. doi: 10.48550/arXiv.2312.04927. URL https://arxiv.org/abs/2312.04927.

Simran Arora, Aman Timalsina, Aaryan Singhal, Benjamin Spector, Sabri Eyuboglu, Xinyi Zhao, Ashish Rao, Atri Rudra, and Christopher Ré. Just read twice: Closing the recall gap for recurrent language models. *arXiv preprint arXiv:2407.05483*, 2024. doi: 10.48550/arXiv.2407.05483. URL https://arxiv.org/abs/2407.05483.

Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, Leo Gao, Jonathan Tow, Baber Abbasi, Alham Fikri Aji, Pawan Sasanka Ammanamanchi, Sidney Black, Jordan Clive, Anthony DiPofi, Julen Etxaniz, Benjamin Fattori, Jessica Zosa Forde, Charles Foster, Jeffrey Hsu, Mimansa Jaiswal, Wilson Y. Lee, Haonan Li, Charles Lovering, Niklas Muennighoff, Ellie Pavlick, Jason Phang, Aviya Skowron, Samson Tan, Xiangru Tang, Kevin A. Wang, Genta Indra Winata, François Yvon, and Andy Zou. Lessons from the trenches on reproducible evaluation of language models. *arXiv preprint arXiv:2405.14782*, 2024. doi: 10.48550/arXiv.2405.14782. URL https://arxiv.org/abs/2405.14782.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020. doi: 10.1609/aaai.v34i05.6239. URL https://ojs.aaai.org/index.php/AAAI/article/view/6239.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*, 2024. URL https://arxiv.org/abs/2406.02069.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3829–3846. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.232. URL https://aclanthology.org/2023.emnlp-main.232/.

Chenglong Chu, Guorui Zhou, Guowang Zhang, Han Li, Hao Peng, Hongtao Cheng, Hui Wang, Jian Liang, Jiangxia Cao, Kun Gai, Lingzhi Zhou, Lu Ren, Qi Zhang, Ruiming Tang, Ruitao Wang, Xinchen Luo, Yi Su, Zhiyuan Liang, Ziqi Wang, Boyang Ding, Chengru Song, Dunju Zang, Jiao Ou, Jiaxin Deng, Jijun Shi, Jinghao Zhang, Junmin Chen, Lejian Ren, Minxuan Lv, Qianqian Wang, Qigen Hu, Shiyao Wang, Siyang Mao, Tao Wang, Xingmei Wang, Zhixin Ling, Ziming Li, and Zixing Zhang. Kwai summary attention technical report. *arXiv preprint arXiv:2604.24432*, 2026. URL https://arxiv.org/abs/2604.24432.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018. doi: 10.48550/arXiv.1803.05457. URL https://arxiv.org/abs/1803.05457.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026. URL https://arxiv.org/abs/2606.19348.

Chenlong Deng, Zhisong Zhang, Kelong Mao, Shuaiyi Li, Tianqing Fang, Hongming Zhang, Haitao Mi, Dong Yu, and Zhicheng Dou. UniGist: Towards general and hardware-aligned sequence-level long context compression. *arXiv preprint arXiv:2509.15763*, 2025. URL https://arxiv.org/abs/2509.15763.

Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, Pradeep Dasigi, Robert Berry, Saumya Malik, Saurabh Shah, Scott Geng, Shane Arora,

Shashank Gupta, Taira Anderson, Teng Xiao, Tyler Murray, Tyler Romero, Victoria Graf, Akari Asai, Akshita Bhagia, Alexander Wettig, Alisa Liu, Aman Rangapur, Chloe Anastasiades, Costa Huang, Dustin Schwenk, Harsh Trivedi, Ian Magnusson, Jaron Lochner, Jiacheng Liu, Lester James V. Miranda, Maarten Sap, Malia Morgan, Michael Schmitz, Michal Guerquin, Michael Wilson, Regan Huff, Ronan Le Bras, Rui Xin, Rulin Shao, Sam Skjonsberg, Shannon Zejiang Shen, Shuyue Stella Li, Tucker Wilde, Valentina Pyatkin, Will Merrill, Yapei Chang, Yuling Gu, Zhiyuan Zeng, Ashish Sabharwal, Luke Zettlemoyer, Pang Wei Koh, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. OLMo 3. *arXiv preprint arXiv:2512.13961*, 2025. URL https://arxiv.org/abs/2512.13961.

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2307.06945.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*, 2024. doi: 10.48550/arXiv.2404.06654. URL https://arxiv.org/abs/2404.06654.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017. doi: 10.18653/v1/D17-1082. URL https://aclanthology.org/D17-1082/.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-0722. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/28ab418242603e0f7323e54185d19bde-Abstract-Conference.html.

Zhixuan Lin, Evgenii Nikishin, Xu Owen He, and Aaron Courville. Forgetting transformer: Softmax attention with a forget gate. In *International Conference on Learning Representations*, 2025. doi: 10.48550/arXiv.2503.02130. URL https://arxiv.org/abs/2503.02130.

Akide Liu, Jing Liu, Zizheng Pan, Yefei He, Gholamreza Haffari, and Bohan Zhuang. MiniCache: KV cache compression in depth dimension for large language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024a. doi: 10.52202/079017-4443. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/fd0705710bf01b88a60a3d479ea341d9-Abstract-Conference.html.

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. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-2279. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/a452a7c6c463e4ae8fbdc614c6e983e6-Abstract-Conference.html.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32332–32344. PMLR, 2024b. URL https://proceedings.mlr.press/v235/liu24bz.html.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1711.05101.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1609.07843.

Amirkeivan Mohtashami and Martin Jaggi. Landmark attention: Random-access infinite context length for transformers. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2305.16300.

Jesse Mu, Xiang Li, and Noah Goodman. Learning to compress prompts with gist tokens. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-0848. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/3d77c6dcc7f143aa2154e7f4d5e22d68-Abstract-Conference.html.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016. doi: 10.18653/v1/P16-1144. URL https://aclanthology.org/P16-1144/.

Qwen. Qwen3-Next-80B-A3B-Instruct model card. Hugging Face, 2025. URL https://huggingface.co/Qwen/Qwen3-Next-80B-A3B-Instruct. Accessed 2026-09-25.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*, 2020. doi: 10.48550/arXiv.1911.05507. URL https://arxiv.org/abs/1911.05507.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020. doi: 10.1609/aaai.v34i05.6399. URL https://ojs.aaai.org/index.php/AAAI/article/view/6399.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/7181-attention-is-all-you-need.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. HuggingFace's transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, 2020. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6/.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2309.17453.

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, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. doi: 10.48550/arXiv.2505.09388. URL https://arxiv.org/abs/2505.09388.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Y. X. Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang,

and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention. *arXiv preprint arXiv:2502.11089*, 2025. URL https://arxiv.org/abs/2502.11089.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-1506. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/6ceefa7b15572587b78ecfcebb2827f8-Abstract-Conference.html.

A NCSA FORWARD PASS

Algorithm 1 gives the forward pass of base NCSA. Chunks follow the boundaries defined in Section 2, while the local and remote index sets $\mathcal{L}_i$ and $\mathcal{R}_i$ follow Eq. 2. The temporary matrices Q_X^ℓ, K_X^ℓ, and V_X^ℓ cache the text projections computed once per layer. Operations within each **for all** block are parallel; the two blocks remain sequential so that text queries read the summaries written in the same layer.

Algorithm 1 NCSA forward pass

Require: Token states $X^0 \in \mathbb{R}^{S\times d}$; chunk size C; visibility sets $\{\mathcal{L}_i, \mathcal{R}_i\}_{i=1}^{S}$; shared anchor a; L Transformer layers
Ensure: Final text states X^L and per-layer compressed KV memories

$G \leftarrow \lceil S/C \rceil$
$z_j^0 \leftarrow a$ for every chunk $j \in \{1, \dots, G\}$
for $\ell = 0, \dots, L-1$ **do**
  $Q_X^\ell \leftarrow q^\ell(X^\ell)$, $K_X^\ell \leftarrow k^\ell(X^\ell)$, $V_X^\ell \leftarrow v^\ell(X^\ell)$
  for all chunks $j \in \{1, \dots, G\}$ **in parallel do**
    $\widetilde{z}_j^\ell \leftarrow z_j^\ell + \mathrm{Attn}\big(q^\ell(z_j^\ell),\ K_X^\ell[(j-1)C+1 : \min(jC, S)],$
      $V_X^\ell[(j-1)C+1 : \min(jC, S)]\big) W_O^\ell$
    $\widetilde{k}_j^\ell \leftarrow k^\ell(\widetilde{z}_j^\ell)$, $\widetilde{v}_j^\ell \leftarrow v^\ell(\widetilde{z}_j^\ell)$ ▷ updated summary KV
    $\widetilde{\mathbf{m}}_j^\ell \leftarrow (\widetilde{k}_j^\ell, \widetilde{v}_j^\ell)$
  end for
  for all text positions $i \in \{1, \dots, S\}$ **in parallel do**
    $\mathcal{K}_i^\ell \leftarrow \left[K_X^\ell[\mathcal{L}_i]; \underset{j\in\mathcal{R}_i}{\mathrm{stack}}\, \widetilde{k}_j^\ell\right]$
    $\mathcal{V}_i^\ell \leftarrow \left[V_X^\ell[\mathcal{L}_i]; \underset{j\in\mathcal{R}_i}{\mathrm{stack}}\, \widetilde{v}_j^\ell\right]$
    $H_x^\ell[i] \leftarrow x_i^\ell + \mathrm{Attn}\big(Q_X^\ell[i], \mathcal{K}_i^\ell, \mathcal{V}_i^\ell\big)\, W_O^\ell$ ▷ same-layer read
    $x_i^{\ell+1} \leftarrow H_x^\ell[i] + \mathrm{FFN}^\ell\Big(\mathrm{Norm}_2^\ell(H_x^\ell[i])\Big)$
  end for
  $z_j^{\ell+1} \leftarrow \widetilde{z}_j^\ell$ for every chunk j ▷ summary bypasses the FFN
  Store $\{\widetilde{\mathbf{m}}_j^\ell\}_{j=1}^{G}$ as the compressed KV memory of layer ℓ
end for
return X^L and $\{\widetilde{\mathbf{m}}_j^\ell : 0 \le \ell < L,\ 1 \le j \le G\}$

B EXPERIMENTAL DETAILS AND HYPERPARAMETERS

B.1 MODEL CONFIGURATIONS

Backbone architecture. All models are trained from random initialization and use a decoder-only, pre-normalization Transformer with the Qwen3 (Yang et al., 2025) block design. The backbone combines grouped-query attention (GQA), per-head QK normalization, rotary position embeddings with base 10^6, RMSNorm with $\epsilon = 10^{-6}$, and SwiGLU feed-forward layers. Input and output embeddings are tied. Dense, CSA, KSA, and NCSA use identical backbone dimensions at each scale; only their attention architecture differs. Table 5 lists the three configurations. The 0.3B model is our reduced configuration rather than an official Qwen3 model size.

Because CSA adds a dedicated compressor, its parameter count is higher than that of the other architectures. KSA adds one vocabulary row for its summary token, whereas NCSA adds one d-dimensional anchor vector; both additions are negligible relative to the backbone. Table 6 reports counts from the evaluated checkpoints. The 0.3B Dense, KSA, and NCSA counts are computed from the verified configuration using the same tied-embedding parameterization as the 0.6B checkpoints.

Table 5: Backbone configurations for the three main model scales. Q/KV gives the numbers of query and key–value heads.

Attribute	0.3B	0.6B	4B
Transformer layers	16	28	36
Hidden dimension	1024	1024	2560
FFN intermediate dimension	2752	3072	9728
Q/KV heads	16/8	16/8	32/8
Head dimension	64	128	128
Normalization	RMSNorm ($\epsilon = 10^{-6}$) with QK normalization		
Position embedding	RoPE ($\theta = 10^6$)		
Activation	SwiGLU		
Context length	4096		
Maximum position embeddings	40,960		

Table 6: Trainable parameter counts for the evaluated checkpoints.

Architecture	0.3B	0.6B	4B
Dense	341,216,256	596,049,920	4,022,468,096
CSA	374,844,672	713,748,928	4,400,288,384
KSA	341,217,280	596,050,944	4,022,470,656
NCSA	341,217,280	596,050,944	4,022,470,656

B.2 Compression configurations

Table 7 summarizes the controls used in the main comparison. All three compressed models group four text tokens into one long-range KV entry and preserve the most recent 128 text tokens as raw local context. Compression is enabled in every Transformer layer. CSA builds its compressed entries with a dedicated per-layer module; in our baseline, all compressed entries are visible to every query, and the DeepSeek-V4 indexer and top-k selector are not used. KSA processes a learned summary token through the standard token path, including the feed-forward network. NCSA instead maintains a summary residual stream across layers and writes a newly formed summary before the same layer's text attention reads it. These settings are fixed across scales unless explicitly varied in an ablation.

Table 7: Attention configurations used in the main experiments.

Attribute	Dense	CSA	KSA	NCSA
Compression ratio	–	4	4	4
Chunk size (tokens)	–	4	4	4
Raw local window (tokens)	Full	128	128	128
Compressed entries per chunk	–	1	1	1
Dedicated compressor	–	Yes	No	No
Summary FFN path	–	–	Yes	No
Same-layer write and read	–	Yes	No	Yes

B.3 Pretraining details

Data. We use the 150B-token sample of the Dolma 3 mixture released by the Allen Institute for AI (AI2) as part of the OLMo 3 project (Ettinger et al., 2025). The source sample contains approximately 157B tokens under the OLMo tokenizer and combines Common Crawl (76.9%), scientific PDFs processed by olmOCR (12.6%), Stack-Edu code (7.1%), FineMath 3+ (2.6%), arXiv (0.8%), and Wikipedia/Wikibooks (0.04%). We use the released corpus as provided, including its existing cleaning, deduplication, and quality filtering, and perform no additional filtering. The documents are

retokenized with the Qwen tokenizer, followed by <|im_end|> as the end-of-document marker, and packed into 4096-token sequences. This conversion produces a 161B-token pool. We sample 50B tokens from its natural mixture, corresponding to approximately 0.31 epochs, without repeating any shard. The dataset release is available at huggingface.co/datasets/allenai/dolma3_mix-150B-1025.

Matched optimization. All four architectures at a given scale use the same tokenized examples in the same order. Concretely, they share the data blend, sequence length, sample count, random seed, and cached dataset indices. Each run processes 12,207,104 sequences with a global batch size of 512 for 23,842 updates, totaling 50,000,297,984 tokens, which we report as 50B. We train in BF16 with Adam, $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, weight decay 0.1, and gradient clipping at 1.0. The learning rate increases linearly to 3×10^{-4} over 2,000 steps and then remains constant. We use seed 1234, set attention and hidden dropout to zero, and select the final 50B-token checkpoint without consulting downstream results.

Table 8: Pretraining hyperparameters. All four attention architectures use the same values.

Attribute	0.3B	0.6B	4B
Training tokens	50B	50B	50B
Sequence length	4096	4096	4096
Global batch (sequences)	512	512	512
Microbatch size per GPU	2	2	1
Training updates	23,842	23,842	23,842
Tokenizer	Qwen2Tokenizer (BBPE)		
Base vocabulary size	151,936		
Pretraining data	Dolma 3 150B sample		
Optimizer	Adam ($\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$)		
Peak / minimum learning rate	3×10^{-4} / 3×10^{-4}		
Learning-rate schedule	2,000-step linear warmup, then constant		
Weight decay / gradient clipping	0.1 / 1.0		
Initialization standard deviation	0.02		
Training precision	BF16		
Attention / hidden dropout	0.0 / 0.0		
Random seed	1234		
Hardware	16–32 NVIDIA Hopper 80GB GPUs		

Training uses tensor, pipeline, and context parallel degrees of one, with sequence parallelism and a distributed optimizer. No activation recomputation is used. Smaller-model runs use 16 or 32 GPUs depending on availability, while all 4B runs use 32 GPUs. Because the global batch, update count, optimizer, and exact sample order remain fixed, this hardware difference does not change the training objective or token budget.

B.4 Evaluation protocol

We evaluate the final 50B-token checkpoint from each run using Transformers 4.57.1 (Wolf et al., 2020) and lm-eval-harness 0.4.12 (Biderman et al., 2024). All evaluations use BF16 inference and batch size one. Results collected by the separate in-training monitors are not included in the reported tables.

Language modeling and general capability. We measure WikiText word perplexity by rolling log-likelihood on the test split of EleutherAI/wikitext_document_level (wikitext-2-raw-v1), with the evaluation length capped at 4096 tokens. On LAMBADA, we report both perplexity and exact-match accuracy. We evaluate HellaSwag, ARC-Easy, ARC-Challenge, PIQA, WinoGrande, and RACE zero-shot using conditional answer likelihood. Following the harness definitions, HellaSwag, both ARC tasks, and PIQA use length-normalized accuracy,

whereas WinoGrande and RACE use raw accuracy. The Avg. column in Table 1 is the mean of these six accuracies and excludes perplexity.

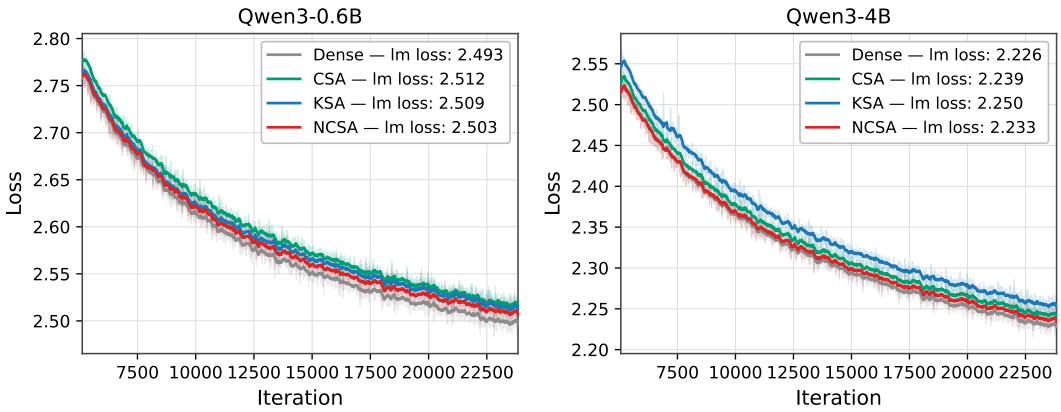
Long-context evaluation. RULER is evaluated at 1K, 2K, 4K, and 8K with 500 examples per task and greedy decoding. Its 13 tasks comprise eight needle-retrieval variants, Variable Tracking (VT), Common Words Extraction (CWE), Frequent Words Extraction (FWE), SQuAD, and HotpotQA. The generation limits are 128 tokens for retrieval, 30 for VT, 120 for CWE, 50 for FWE, and 32 for each QA task. The two QA tasks are excluded at 1K and 2K because their source documents do not fit, giving an 11-task average at these lengths and a 13-task average at 4K and 8K. The 8K evaluation uses the same checkpoints pretrained at 4K and therefore measures length extrapolation. Table 2 reports capability-level aggregates and the official mean; Appendix D provides the complete 4K results.

In-context retrieval. We use the official JRT `based_*` tasks at a 2048-token context: SWDE (1,067 examples), FDA (1,101), SQuAD (2,977), TriviaQA (1,688), Natural Questions (3,157), and DROP (2,087). All models are base checkpoints without instruction tuning. We use the official context construction and truncation, reserve 50 answer tokens, decode greedily, and report the case-insensitive `contains` score.

Diagnostic evaluations. The single-layer MQAR probe and the position-wise PG19 analysis use custom controlled protocols. Their complete task construction, optimization, sampling, and smoothing settings are provided in Appendices G and H.

C Training Curves and Commonsense Results

Figure 4 reports matched-data training curves at 0.6B and 4B, omitted from the main text. The curves are directly comparable step by step because architectures at the same scale consume the same training examples in the same order.



(a) Training loss, 0.6B.

(b) Training loss, 4B.

Figure 4: Matched-data pretraining loss of the four architectures. **(a,b)** show the 0.6B and 4B training-loss curves, respectively.

D Complete 4K RULER results

E JRT In-Context Retrieval Details

We follow the official Just Read Twice (JRT) in-context retrieval implementation and use its `based_*` task variants rather than the standard closed-book or instruction-formatted tasks in `lm-eval`. All evaluations use a 2048-token context, 50 reserved answer tokens, greedy decoding, BF16, batch size one, and no few-shot demonstrations. Context construction, truncation, prompts,

Table 9: Complete per-task RULER results at 4K context, reported as percentages. S-1–S-3 are Single-NIAH variants; MQ and MV are Multi-Query and Multi-Value NIAH; MK-1–MK-3 are Multi-Key NIAH variants. VT, CWE, FWE, SQD, and HPT denote Variable Tracking, Common Words Extraction, Frequent Words Extraction, SQuAD, and HotpotQA.

Scale	Architecture	Retrieval								VT	Aggregation		QA		Avg13
		S-1	S-2	S-3	MQ	MV	MK-1	MK-2	MK-3		CWE	FWE	SQD	HPT	
0.3B	Dense	99.80	99.40	94.80	27.75	27.95	59.00	7.40	3.80	1.40	23.34	0.53	27.87	22.80	38.14
	CSA	88.20	77.00	17.40	19.70	18.35	50.40	1.60	0.00	0.28	0.16	0.00	11.53	16.00	23.12
	KSA	97.20	89.00	59.80	29.50	28.30	46.00	1.60	0.60	6.60	22.48	3.53	19.92	18.40	32.53
	NCSA	98.40	94.80	80.60	25.45	21.55	47.80	3.20	1.40	10.76	22.42	1.67	20.20	22.60	34.68
0.6B	Dense	100.00	100.00	94.60	51.15	34.35	82.40	25.80	7.60	6.16	37.32	32.20	36.42	28.80	48.98
	CSA	92.20	88.60	60.00	22.90	25.62	40.80	5.60	0.20	1.64	25.62	34.13	22.30	21.00	33.89
	KSA	99.20	98.80	97.20	24.70	19.05	40.60	9.40	2.00	1.08	31.16	31.07	30.98	22.80	39.08
	NCSA	100.00	99.60	96.00	38.25	30.35	62.60	9.20	1.20	6.84	32.14	30.20	30.12	25.00	43.19
4B	Dense	100.00	100.00	85.20	74.25	48.05	81.80	64.80	24.60	16.88	45.70	45.07	46.18	33.60	58.93
	CSA	92.20	97.40	87.40	48.70	43.60	62.00	15.20	14.40	9.88	34.32	40.80	36.90	33.00	47.37
	KSA	100.00	100.00	96.60	72.20	65.60	73.00	47.20	5.60	2.28	36.62	15.73	42.87	31.00	52.98
	NCSA	99.40	100.00	96.80	70.35	65.60	83.40	30.20	18.20	22.12	59.40	49.33	45.82	33.00	59.51

and metrics are unchanged. Table 10 reports the official case-insensitive `contains` score. Natural Questions follows JRT's `based_nq_2048` split; its official truncation can remove the gold evidence for some examples, which partly explains its lower absolute score.

Table 10: Complete JRT in-context retrieval results at 2K context (%). Avg. is the mean of all six tasks. Best and second-best results at each scale are shown in **bold** and underlined.

Scale	Model	SWDE	FDA	SQD	TQA	NQ	DROP	Avg.
0.3B	Dense	40.67	**63.94**	**36.11**	45.56	**20.53**	22.23	**38.17**
	CSA	29.52	32.43	30.60	40.82	14.35	**22.47**	28.36
	KSA	32.99	40.51	32.85	44.31	18.15	22.33	31.86
	NCSA	**41.80**	49.50	31.98	**45.85**	19.10	20.17	34.73
0.6B	Dense	**52.67**	**67.94**	**39.54**	52.78	**26.07**	23.81	**43.80**
	CSA	43.30	57.86	37.05	50.12	22.33	**23.91**	39.09
	KSA	48.27	56.58	36.92	51.30	23.85	23.14	40.01
	NCSA	45.27	62.67	35.91	**53.32**	23.95	23.57	40.78
4B	Dense	**63.36**	**75.30**	**45.78**	**63.86**	**31.52**	**29.90**	**51.62**
	CSA	54.83	65.40	42.79	60.01	29.01	24.72	46.13
	KSA	57.54	73.84	45.25	60.96	29.74	27.93	49.21
	NCSA	60.36	73.30	43.57	59.42	31.39	29.56	49.60

F Complete Architecture Ablation Results

Table 11 reports the full 0.6B results summarized in Table 4. Dense is included as a matched non-compressed reference; the remaining rows use the NCSA pathway or one of its architectural variants.

G Single-Layer MQAR Probe

To isolate whether input-dependent content can enter the remote channel within one layer, we use a single-layer MQAR probe. The corresponding learning curves are shown in Figure 5.

Each 256-token MQAR sequence contains four key–value pairs placed in 8-token chunks, with two pairs per chunk. Keys, values, and filler symbols are sampled from disjoint portions of a 194-symbol

Table 11: Complete architecture-ablation results at 0.6B after 50B training tokens. Commonsense tasks use zero-shot accuracy (%); Wiki and LAMB. report perplexity (lower is better); RULER tasks are evaluated at 4K (%).

	Commonsense reasoning							Language modeling		RULER at 4K											
Variant	Hella.	ARC-E	ARC-C	PIQA	Wino.	RACE	Avg6	Wiki	LAMB.	S1	S2	S3	MK1	MK2	MK3	MQ	MV	VT	CWE	FWE	SQD	HPT	Avg13
Dense	43.07	55.26	29.86	67.30	53.43	32.25	46.86	21.62	16.69	100.00	100.00	94.60	82.40	25.80	7.60	51.15	34.35	6.16	37.32	32.20	36.42	28.80	48.98
NCSA	43.37	57.20	28.84	67.57	54.38	32.73	47.35	21.94	15.40	100.00	99.60	96.00	62.60	9.20	1.20	38.25	30.35	6.84	32.14	30.20	30.12	25.00	43.19
No stream	41.33	56.65	28.33	66.32	53.91	31.20	46.29	22.90	18.00	99.20	99.60	88.80	58.00	19.60	0.40	26.20	25.05	5.88	36.84	35.80	28.77	26.80	42.38
No residual	42.93	57.32	28.50	67.90	50.91	31.87	46.57	22.00	14.96	98.60	97.00	92.40	44.60	10.60	4.20	29.95	26.85	10.12	31.02	34.07	34.78	24.40	41.43
Summary FFN	42.58	53.58	28.50	67.95	51.93	31.67	46.04	21.98	15.67	100.00	99.80	92.80	70.20	16.00	9.60	32.60	26.10	13.24	42.14	30.67	28.85	25.00	45.15
3:1 hybrid	42.93	55.56	28.67	68.50	52.17	31.20	46.51	21.47	15.81	100.00	100.00	94.40	67.20	3.20	3.00	59.70	56.50	0.68	43.52	46.80	32.05	26.00	48.70

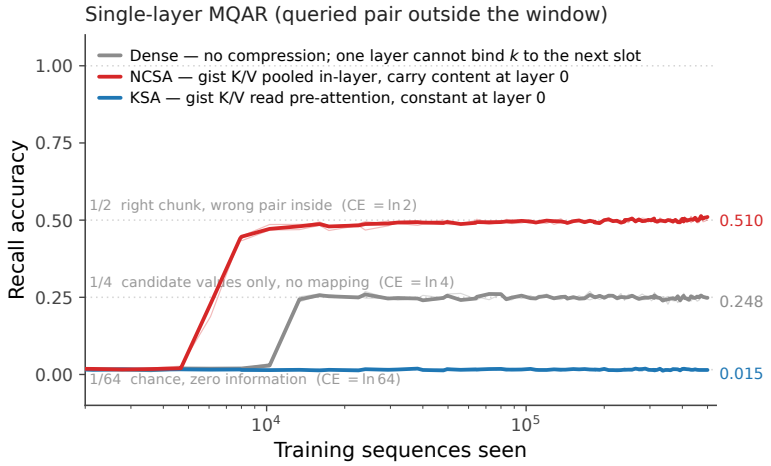


Figure 5: **Single-layer MQAR.** Recall accuracy versus online-generated training sequences. Thick curves are medians over three seeds at the better of two learning rates; thin curves show individual seeds. KSA remains at chance ($1/64$), Dense reaches $1/4$, and NCSA reaches approximately $1/2$.

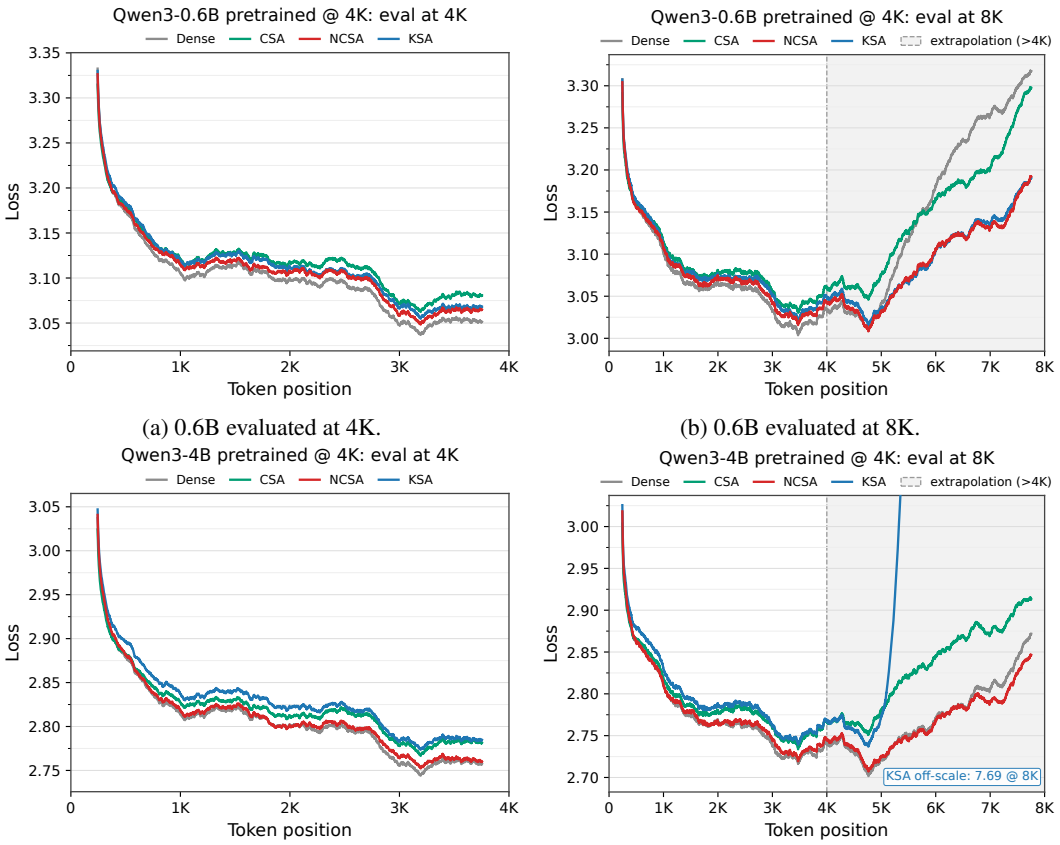
vocabulary. The four keys reappear in random order in a query region at the end, but their values are not repeated there. We force every queried pair outside the visible raw-text window, so success requires the remote channel. Loss and accuracy are computed only where the queried values should be predicted.

Dense, KSA, and NCSA use the same single-layer decoder with hidden size 128, four attention heads, and a SwiGLU feed-forward block. Models are trained with AdamW (Loshchilov & Hutter, 2019) on 500K online-generated sequences; data are not repeated. We sweep learning rates $\{10^{-3}, 3 \times 10^{-3}\}$ and run three seeds per setting. Figure 5 reports the median at the better learning rate and shows individual seeds. At the final budget, KSA reaches 0.0146 accuracy with cross-entropy 4.159 ($\ln 64$), Dense reaches 0.2485 with loss 1.391 ($\ln 4$), and NCSA reaches 0.5103 with loss 0.697 ($\ln 2$). A separate parity check confirms that layer-0 KSA summary KV is identical across prompts, whereas NCSA's in-layer summary KV depends on chunk content.

H Position-Wise Loss Details

We tokenize PG19 test books into a continuous stream and split it into non-overlapping windows. Teacher-forced cross-entropy at each text position is averaged over 300 windows: 1.23M evaluated tokens at 4K and 2.46M at 8K. Curves use a running mean of width 501. The four architectures are evaluated at identical text positions; their internal summary states do not add positions to the reported loss. Because all checkpoints are pretrained at 4K, positions 4K–8K measure extrapolation.

Within 4K, the full-sequence mean losses differ by at most 0.016 nats at 0.6B and 0.022 nats at 4B. At 0.6B, both summary-based architectures remain nearly flat from the first to the second half of the 8K windows. At 4B, NCSA remains stable (2.800 to 2.771), Dense is comparable (2.797 to 2.778), CSA rises from 2.809 to 2.837, and KSA rises from 2.818 to 5.007.



(a) 0.6B evaluated at 4K.

(b) 0.6B evaluated at 8K.

(c) 4B evaluated at 4K.

(d) 4B evaluated at 8K.

Figure 6: Complete position-wise loss results on PG19. All checkpoints are pretrained with a 4K sequence length; shaded regions in the 8K panels denote extrapolation.

Within 4K, the full-sequence mean losses differ by at most 0.016 nats at 0.6B and 0.022 nats at 4B. At 0.6B, both summary-based architectures remain nearly flat from the first to the second half of the 8K windows. At 4B, NCSA remains stable (2.800 to 2.771), Dense is comparable (2.797 to 2.778), CSA rises from 2.809 to 2.837, and KSA rises from 2.818 to 5.007.