CoWindow Attention: Full Causal Coverage Is a Collective Property

Anonymous authors
Paper under double-blind review

Abstract

Long-context full attention (FullAttn) repeatedly exposes the complete causal history to every attention head, creating substantial redundant computation and memory traffic even with IO-efficient dense kernels. We introduce **CoWindow Attention (CoWA)**, a structured attention architecture that distributes access to the causal history across KV heads. All heads share near-diagonal and prefix-sink windows, while complementary long-range windows partition the remaining history. Their union provides full causal coverage although each head attends sparsely to distant tokens. This position-defined attention pattern requires no learned router or indexer, is used consistently during training and inference, and aligns with KV-head tensor parallelism. A window-matched ablation at 8K isolates the effect of complementary long-range allocation: CoWA with 100% collective coverage reaches 89.73% accuracy, compared with 89.97% for FullAttn, while duplicated long-range windows perform substantially worse. Across a broader controlled associative-recall comparison with matched token budgets, CoWA closely tracks FullAttn as the context grows, whereas other sparse patterns lose a substantial fraction of the associations. In an attention-operator benchmark at 128K tokens on 8 H100 GPUs with tensor parallelism, CoWA reduces forward and backward latency during training by $7.4\times$ and $8.6\times$ and decoding latency during inference by $3.0\times$ over FullAttn. Its per-rank peak operator memory matches FullAttn during training and is $7.6\times$ lower during decoding. Across scaling-law training from 0.6B to 14B parameters on 128 H100 GPUs, CoWA closely tracks FullAttn in perplexity while reducing total training FLOPs, with a 28.5% reduction at 14B during 32K-context training. The resulting 14B models and 32B models from separate continued training achieve comparable knowledge, reasoning, and long-context retrieval scores to FullAttn. These results show that full causal coverage can be a collective property of the head ensemble rather than a duplicated property of every head.

1 Introduction

Context lengths are expanding from thousands to hundreds of thousands of tokens (Snell et al., 2024) to support long-document understanding (Park et al., 2023; DeepMind, 2025), multi-turn reasoning (HuggingFace, 2025; Guo et al., 2025; Team, 2025), and repository-level code generation (Zhang et al., 2024). Attention over these long contexts incurs substantial computation and memory traffic. FlashAttention (Dao et al., 2022; Shah et al., 2024) improves the IO efficiency of self-attention (Vaswani et al., 2017) through tiling, fusion, and online softmax (Milakov & Gimelshein, 2018). However, these IO improvements leave the full attention (FullAttn) pattern unchanged: every head still attends to the full causal prefix.

Retaining direct access to this history within an attention layer does not require every head to access every historical position (Zhao et al., 2025a). For each query, such access can be provided collectively, with every causal token position visible to at least one head. Attention heads develop heterogeneous roles and exhibit distinct long-range and local behaviors (Guo et al., 2024; Gu et al., 2024; Barbero et al., 2025; Xiao et al., 2024b; Sandoval-Segura et al., 2026; Fu et al., 2026). These observations motivate keeping recent context visible to all heads while dividing distant context among heads with complementary receptive fields.

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.

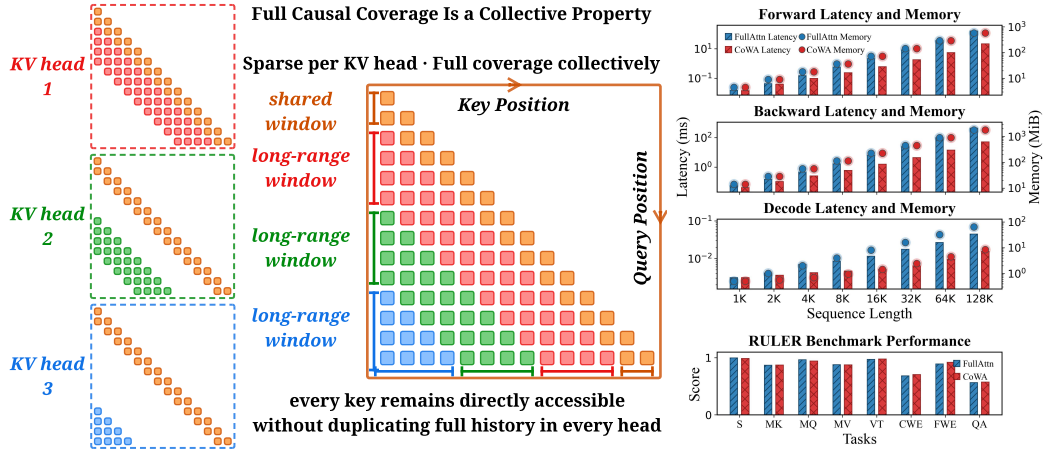

Figure 1: **CoWindow Attention Overview. Left:** All KV heads share the near window and receive complementary long-range windows whose union covers the full causal attention region, preserving direct access to every preceding key position without duplicating the full history in every head. The shared sink window is omitted for visual clarity. **Right:** As sequence length grows, CoWA reduces forward and backward latency during training and decoding latency during inference, while matching FullAttn's peak operator memory during training and reducing it during decoding. Separate model-level evaluations show comparable performance on the RULER benchmark.

Coordinating this access across KV heads also requires an execution structure that remains efficient during training and inference. Sliding-window attention is simple and efficient, but applies the *same finite horizon* to each head and removes direct access to more distant tokens (Fu et al., 2025). Global tokens and fixed block patterns retain only predetermined long-range paths (Child et al., 2019; Zaheer et al., 2020), while dynamic approaches select tokens or blocks through content-dependent scores or routing (Tang et al., 2024; Lai et al., 2025; Li et al., 2024; Zhang et al., 2023; Xiao et al., 2024a; Qi et al., 2026; Zhao et al., 2025b; Yuan et al., 2025; Lu et al., 2025; Gao et al., 2024). These dynamic mechanisms offer greater flexibility, but introduce a separate selection problem and often depart from the regular window structure that makes sparse attention inexpensive. This raises the question: can distributing access to the causal history across KV heads preserve model quality and long-range retrieval while reducing the cost of training and inference?

We introduce **CoWindow Attention (CoWA)**, which distributes access to the causal history across KV heads, as illustrated in Figure 1. All KV heads share a near-diagonal window that preserves direct access to recent context and a prefix-sink window that keeps the beginning of the sequence visible. CoWA then partitions the remaining causal distances into complementary long-range windows and assigns one window to each KV head. Consequently, each head remains locally dense but becomes sparse over the distant history, while together the heads can attend to every position in the causal prefix. We call this property *collective coverage*.

CoWA preserves full causal coverage within each attention layer while reducing duplicated long-range query-key connections across heads. We train the input projections and output projection end to end under this sparse, position-defined attention pattern, allowing the model to learn how to combine information accessed by different heads.

The same decomposition also yields a regular execution structure. Each head is described by a small set of contiguous windows, allowing excluded attention blocks to be omitted directly rather than discovered through a dense mask or a separate router. A single position-defined window-construction rule governs the training forward and backward passes, inference prefill and autoregressive decoding. Global KV-head indexing preserves complementary window assignments across tensor-parallel ranks.

We evaluate CoWA around two questions: whether collective coverage preserves language-model quality and long-range retrieval when no individual head observes the full history, and whether the resulting reduction in duplicated long-range access translates into practical training and inference efficiency under tensor parallelism (Shoeybi et al., 2019). A window-matched collective-coverage ablation at 8K first isolates the effect of complementary long-range allocation: CoWA reaches 89.73% accuracy, compared with 89.97% for FullAttn, while duplicated long-range windows perform

substantially worse under matched per-head window widths. Across a broader controlled associative-recall comparison with matched token budgets among sparse methods, CoWA closely tracks FullAttn as the context grows, whereas the other sparse patterns degrade substantially. In the attention-operator benchmark at 128K sequence length on 8 H100 GPUs with tensor parallelism, CoWA reduces forward and backward latency during training by $7.4\times$ and $8.6\times$, and decoding latency during inference by $3.0\times$ relative to FullAttn. Across scaling-law training from 0.6B to 14B parameters, CoWA closely tracks FullAttn in perplexity; at 14B, it reduces total training FLOPs by 3.1% during 4K pre-training and 28.5% during 32K long-context training. At the model level, the resulting 14B models and separately continued-trained 32B models retain aggregate knowledge, reasoning, and long-context retrieval performance comparable to FullAttn. Our contributions are as follows:

- We formulate **CoWindow Attention**, which distributes access to the causal history through shared near and sink windows and complementary long-range windows. This construction provides full causal coverage while reducing duplicated long-range access across heads.
- We implement CoWA for training and inference using a position-defined window rule aligned with KV-head tensor parallelism.
- We evaluate the modeling and efficiency consequences of collective coverage across long-context language modeling, retrieval, kernel execution, and distributed deployment, and isolate its contribution through a window-matched ablation.

2 METHODOLOGY

We now formalize the head-wise attention pattern underlying CoWA and describe its realization for training and inference under tensor parallelism.

2.1 COWINDOW ATTENTION PATTERN

The default CoWA pattern assigns each KV head three windows: shared near-diagonal and prefix-sink windows, and a head-specific long-range window. The long-range windows cover complementary causal-distance intervals across KV heads.

Let N_q and N_k denote the query and key sequence lengths, with zero-based token indices $0 \le i_q < N_q$ and $0 \le i_k < N_k$ throughout. To use one definition throughout training and inference, including variable-length prefill and autoregressive decoding, we define the bottom-right-aligned causal distance

$$\delta(i_q, i_k) := i_q + (N_k - N_q) - i_k \tag{1}$$

A key is causal for query position i_q exactly when $\delta(i_q, i_k) \ge 0$. When $N_q = N_k$, this reduces to the usual distance $i_q - i_k$.

Let H_q and H_k be the numbers of query/output (QO) and key/value (KV) heads, with H_q an integer multiple of H_k. Under grouped-query attention (Ainslie et al., 2023), QO head h uses the key and value tensors and the window assignment of KV head $h_k = \lfloor hH_k/H_q \rfloor$. The associated H_q/H_k QO heads form a QO-head group that shares keys, values, and the visible-key set while computing attention weights from distinct queries. We first describe the default pattern, in which all KV heads share a prefix-sink window of width w_{sink} and a near-diagonal window of width w_{near}. The remaining long-range span is divided evenly using breakpoints b_{h_k}:

$$L := \max(N_k - w_{\text{sink}} - w_{\text{near}}, 0) \qquad b_{h_k} := \lfloor h_k L / H_k \rfloor, \quad h_k = 0, \dots, H_k \tag{2}$$

For a fixed query i_q, write $\delta = \delta(i_q, i_k)$. The three windows assigned to KV head h_k are

$$\begin{aligned}
\mathcal{W}^{\text{near}}_{h_k}(i_q) &:= \{i_k : 0 \le \delta < w_{\text{near}}\} && \text{(near-diagonal)} \\
\mathcal{W}^{\text{sink}}_{h_k}(i_q) &:= \{i_k : 0 \le i_k < w_{\text{sink}}, \delta \ge 0\} && \text{(prefix-sink)} \\
\mathcal{W}^{\text{long}}_{h_k}(i_q) &:= \{i_k : w_{\text{near}} + b_{h_k} \le \delta < w_{\text{near}} + b_{h_k+1}\} && \text{(long-range)}
\end{aligned}$$

Their union gives the visible-key set:

$$\mathcal{A}_{h_k}(i_q) := \mathcal{W}^{\text{near}}_{h_k}(i_q) \cup \mathcal{W}^{\text{sink}}_{h_k}(i_q) \cup \mathcal{W}^{\text{long}}_{h_k}(i_q) \tag{3}$$

Algorithm 1 CoWindow Attention Forward

Require: Matrices $\mathbf{Q}, \mathbf{O} \in \mathbb{R}^{H_q \times N_q \times d_h}$ and $\mathbf{K}, \mathbf{V} \in \mathbb{R}^{H_k \times N_k \times d_h}$. Vector $\ell \in \mathbb{R}^{H_q \times N_q}$. Windows w.

Divide $\mathbf{Q}, \mathbf{O}$ into QO blocks of size B_q and $\mathbf{K}, \mathbf{V}$ into KV blocks of size B_k.
for each QO head $h \in [0, \ldots, H_q - 1]$ in parallel **do**
  Compute $h_k = \lfloor h \times H_k / H_q \rfloor$ under grouped query attention.
  for each QO block I_i in parallel **do**
    Initialize $\mathbf{O}_{hi} = (0) \in \mathbb{R}^{B_q \times d_h}$, $\ell_{hi} = (0) \in \mathbb{R}^{B_q}$, and $\mathbf{m}_{hi} = (-\infty) \in \mathbb{R}^{B_q}$.
    Load $\mathbf{Q}_{hi}, w_{h_k}$.
    Compute the KV-block set $\mathcal{J}_{h_k}(i)$ from w_{h_k}.
    for j descending over $\mathcal{J}_{h_k}(i)$ **do**
      Load $\mathbf{K}_{h_k j}, \mathbf{V}_{h_k j}$.
      Compute $\mathbf{S}_{hij} = \mathbf{Q}_{hi} \mathbf{K}_{h_k j}^\top \in \mathbb{R}^{B_q \times B_k}$ and apply causal or window mask to $\mathbf{S}_{hij}$ at boundaries.
      Compute $\tilde{m}_{hij} = \mathrm{rowmax}(\mathbf{S}_{hij}) \in \mathbb{R}^{B_q}$ and $\mathbf{m}_{hi}^{\mathrm{new}} = \max(\mathbf{m}_{hi}, \tilde{m}_{hij})$.
      Compute $\tilde{\mathbf{P}}_{hij} = \exp(\mathbf{S}_{hij} - \mathbf{m}_{hi}^{\mathrm{new}}) \in \mathbb{R}^{B_q \times B_k}$.
      Compute $\tilde{\ell}_{hij} = \mathrm{rowsum}(\tilde{\mathbf{P}}_{hij}) \in \mathbb{R}^{B_q}$ and $\ell_{hi}^{\mathrm{new}} = \exp(\mathbf{m}_{hi} - \mathbf{m}_{hi}^{\mathrm{new}}) \ell_{hi} + \tilde{\ell}_{hij} \in \mathbb{R}^{B_q}$.
      Update $\mathbf{O}_{hi} \leftarrow \exp(\mathbf{m}_{hi} - \mathbf{m}_{hi}^{\mathrm{new}}) \mathbf{O}_{hi} + \tilde{\mathbf{P}}_{hij} \mathbf{V}_{h_k j} \in \mathbb{R}^{B_q \times d_h}$.
      Update $\ell_{hi} \leftarrow \ell_{hi}^{\mathrm{new}}$ and $\mathbf{m}_{hi} \leftarrow \mathbf{m}_{hi}^{\mathrm{new}}$.
    end for
    Update $\mathbf{O}_{hi} \leftarrow \mathrm{diag}(\ell_{hi})^{-1} \mathbf{O}_{hi}$ and $\ell_{hi} \leftarrow \mathbf{m}_{hi} + \log(\ell_{hi})$.
    Store $\mathbf{O}_{hi}, \ell_{hi}$.
  end for
end for
Return $\mathbf{O}, \ell$.

Each head therefore sees the shared recent context and prefix sinks together with one long-range interval.

Algorithms 1 and 2 represent each head's windows by a four-tuple of sink width, near width, gap after the near window, and long-range width. For the default pattern, this tuple is $w_{h_k} = (w_{\mathrm{sink}}, w_{\mathrm{near}}, b_{h_k}, b_{h_k+1} - b_{h_k})$. The same interface permits head-specific window widths and gaps. Other breakpoint schedules redistribute long-range work under the attention pattern in Equation 3. For load-balanced decoding, we use the default equal-width allocation; equal-area allocation for training is outside the scope of this work.

2.2 Coverage and Attention Cost

Collective coverage. Under the default allocation, all KV heads share the near and sink windows, while adjacent long-range intervals provide access to the remaining causal distances. For each query, the union of these visible-key sets covers the full causal prefix:

$$\bigcup_{h_k=0}^{H_k-1} \mathcal{A}_{h_k}(i_q) = \{i_k \in \{0, \ldots, N_k - 1\} : \delta(i_q, i_k) \geq 0\} \tag{4}$$

Full causal coverage guarantees direct access to token positions, but does not imply the same head-specific interactions or outputs as FullAttn. Each QO head applies softmax over its own visible-key set. This union counts each accessible key once; the attention workload also depends on how many heads include that key.

Attention cost. Consider the final query position $i_q = N_q - 1$ and the common regime $w_{\mathrm{sink}} + w_{\mathrm{near}} \leq N_k$. The sink, near, and long-range regions are disjoint at this position, so the total number of visible keys summed across KV heads is

$$\sum_{h_k=0}^{H_k-1} |\mathcal{A}_{h_k}(N_q - 1)| = H_k(w_{\mathrm{sink}} + w_{\mathrm{near}}) + L \tag{5}$$

For this query, the shared windows contribute once per KV head, while each long-range key belongs to one KV-head support set. Dividing by H_k gives the average number of visible keys per KV head. Dense causal attention has a summed count of $H_k N_k$. With equally sized query groups, multiplying either summed count by H_q / H_k gives the actual number of QO-head query-key connections.

Algorithm 2 CoWindow Attention Backward

Require: Matrices $\mathbf{Q}, \mathbf{O}, \mathbf{dQ}, \mathbf{dO} \in \mathbb{R}^{H_q \times N_q \times d_h}$ and $\mathbf{K}, \mathbf{V}, \mathbf{dK}, \mathbf{dV} \in \mathbb{R}^{H_k \times N_k \times d_h}$. Vector $\ell \in \mathbb{R}^{H_q \times N_q}$. Windows w.

Divide $\mathbf{Q}, \mathbf{O}, \mathbf{dQ}, \mathbf{dO}$ into QO blocks of size B_q and $\mathbf{K}, \mathbf{V}, \mathbf{dK}, \mathbf{dV}$ into KV blocks of size B_k.
for each QO head $h \in [0, \ldots, H_q - 1]$ in parallel **do**
  Compute $h_k = \lfloor h \times H_k / H_q \rfloor$ under grouped query attention.
  for each KV block J_j in parallel **do**
    Initialize $\mathbf{dK}_{h_k j} = (0) \in \mathbb{R}^{B_k \times d_h}$ and $\mathbf{dV}_{h_k j} = (0) \in \mathbb{R}^{B_k \times d_h}$.
    Load $\mathbf{K}_{h_k j}, \mathbf{V}_{h_k j}, w_{h_k}$.
    Compute the QO-block set $\mathcal{I}_{h_k}(j)$ from w_{h_k}.
    for i ascending over $\mathcal{I}_{h_k}(j)$ **do**
      Load $\mathbf{Q}_{hi}, \mathbf{O}_{hi}, \mathbf{dO}_{hi}$.
      Compute $\mathbf{S}_{hji} = \mathbf{K}_{h_k j} \mathbf{Q}_{hi}^\top \in \mathbb{R}^{B_k \times B_q}$ and apply causal or window mask to $\mathbf{S}_{hji}$ at boundaries.
      Compute $\mathbf{P}_{hji} = \exp(\mathbf{S}_{hji} - \ell_{hi}^\top) \in \mathbb{R}^{B_k \times B_q}$ and $\mathbf{dP}_{hji} = \mathbf{V}_{h_k j} \mathbf{dO}_{hi}^\top \in \mathbb{R}^{B_k \times B_q}$.
      Compute $\mathbf{dS}_{hji} = \mathbf{P}_{hji} \circ (\mathbf{dP}_{hji} - \mathrm{rowsum}(\mathbf{O}_{hi} \circ \mathbf{dO}_{hi})^\top) \in \mathbb{R}^{B_k \times B_q}$.
      Update $\mathbf{dV}_{h_k j} \leftarrow \mathbf{dV}_{h_k j} + \mathbf{P}_{hji} \mathbf{dO}_{hi}$ and $\mathbf{dK}_{h_k j} \leftarrow \mathbf{dK}_{h_k j} + \mathbf{dS}_{hji} \mathbf{Q}_{hi}$.
      AtomicAdd $\mathbf{dQ}_{hi} \leftarrow \mathbf{dQ}_{hi} + \mathbf{dS}_{hji}^\top \mathbf{K}_{h_k j}$.
    end for
    AtomicAdd $\mathbf{dK}_{h_k j}, \mathbf{dV}_{h_k j}$.
  end for
end for
Return $\mathbf{dQ}, \mathbf{dK}, \mathbf{dV}$.

The final-query count also applies to each autoregressive decoding step. Full-sequence training accumulates the support counts over all query positions; causal clipping and overlap can reduce the count for earlier queries. The resulting work can differ across heads even when their long-range intervals have equal widths. If $w_{\text{sink}} + w_{\text{near}} > N_k$, the near and sink regions already cover the causal prefix and the number of visible keys per KV head is capped by N_k. For fixed H_k, full-sequence attention remains quadratic in sequence length, with savings arising from fewer duplicated query-key connections. Cross-head decomposition requires $H_k > 1$; when $H_k = 1$, full coverage reduces to dense causal attention.

2.3 Execution and Parallelism

Global KV-head assignment. CoWA assigns windows by global KV-head index so that tensor-parallel ranks receive complementary long-range windows. Let TP denote the number of tensor-parallel ranks, with $TP \le H_k$, and suppose the H_k KV heads are evenly and contiguously sharded across them. For local head h_{local} on rank r, we define

$$h_{\text{global}} = r H_k^{\text{local}} + h_{\text{local}} \qquad H_k^{\text{local}} = H_k / TP \tag{6}$$

The breakpoints in Equation 2 are evaluated using h_{global} and the global H_k throughout training and inference.

Forward execution and parallelism. The same forward operator serves the training forward pass and inference prefill. The QO and KV tensors are stored in head-major order, with shapes given in Algorithm 1. For notational simplicity, $\mathbf{Q}$ is assumed to be pre-scaled by the standard factor $1/\sqrt{d_h}$, where d_h is the head dimension. We divide their token dimensions into QO blocks I_i and KV blocks J_j of sizes B_q and B_k. For QO block I_i, let $\mathcal{J}_{h_k}(i)$ contain the KV blocks with at least one legal pair under Equation 3. Algorithm 1 parallelizes over QO heads and QO blocks and visits only $\mathcal{J}_{h_k}(i)$. Blocks outside this set are skipped before QK computation; only blocks crossing a causal, window, or sequence boundary require elementwise masking.

Backward execution and parallelism. For KV block J_j, let $\mathcal{I}_{h_k}(j)$ contain the QO blocks with at least one legal pair with J_j. Algorithm 2 parallelizes over QO heads and KV blocks and traverses this inverse relation using the same visible-key set as the forward pass. Probabilities reconstructed from the saved logsumexp ℓ therefore yield the exact gradient of the CoWA operator. Atomic additions combine contributions to each QO gradient from multiple KV blocks and to each KV gradient from QO heads in the same group. Under tensor parallelism, each rank applies this traversal to its local KV heads, and their global indices coordinate the sparse pattern across ranks.

Table 1: **Collective Coverage with Matched Per-Head Window Widths.** The 1-, 2-, 4-, and 8-window layouts use matched per-head window widths and vary the number of distinct long-range windows assigned across eight KV heads. FullAttn and SWA are dense and local references and are not part of this window-matched comparison. We report associative-recall accuracy (%) at $d_{model} = 512$.

LAYOUT	WINDOW ASSIGNMENT	LONG-RANGE COVERAGE	1K	2K	4K	8K
FullAttn	Full history per head	100%	100	100	100	89.97
SWA	No long-range window	0%	100	17.48	14.12	5.34
1 unique window	1 window × 8 heads	12.5%	100	53.53	32.99	21.32
2 unique windows	2 windows × 4 heads	25%	100	62.74	40.37	32.92
4 unique windows	4 windows × 2 heads	50%	100	83.91	61.75	52.32
CoWA	8 windows × 1 head	100%	100	100	100	89.73

Decoding execution and parallelism. Autoregressive decoding is the forward case with $N_q = 1$, for which $\delta(0, i_k) = N_k - 1 - i_k$ and the near window contains the latest KV-cache entries. Here, N_k equals the number of previously cached tokens plus one for the current token. At each decoding step, the breakpoints are updated from this current key length, maintaining equal-width long-range windows as the sequence grows. The visible KV blocks can be distributed across split-KV programs and combined through standard partial online-softmax states. Split scheduling controls parallelism and load balance under the same position-defined window rule. Across tensor-parallel ranks, global KV-head indexing preserves the allocation of complementary long-range windows, and each rank stores only its local KV heads. KV-head sharding partitions the cache across ranks, while visiting only visible KV blocks reduces decoding-operator working memory. During decoding in our model-level evaluations, each rank computes QKV projections for the current token and retains the full KV history of its local KV heads in HBM. Window-aware offloading and prefetching to reduce HBM-resident KV storage remain future work. The operator-memory measurements in Section 3 are distinct from persistent KV-cache storage. Configurations with $TP > H_k$ replicate logical KV heads and generally store more than a $1/TP$ fraction of the global KV cache per rank.

3 EXPERIMENTS

Evaluation Overview. The experiments test whether distributing long-range access across heads preserves retrieval accuracy, reduces attention-operator cost, and maintains language-model quality across model scales. We first isolate the role of collective coverage through a window-matched ablation, and then compare a broad set of attention mechanisms on associative recall using randomized key-value bindings. We then benchmark the end-to-end attention-operator costs of the training forward and backward passes and autoregressive decoding for the dense reference and representative trainable sparse methods. Finally, we study scaling behavior from 0.6B to 14B parameters and evaluate the resulting 14B models together with 32B models obtained through a separate continued-training experiment on knowledge, reasoning, and long-context retrieval benchmarks.

Experimental Settings. Within each experiment, attention variants use matched model scales, depth, hidden size, data, and optimization settings, while retaining the head configuration and sparse-selection mechanism of each method. The scaling-law and continued-training experiments are conducted on 128 NVIDIA H100 GPUs, while downstream evaluation and operator benchmarking use 8 H100 GPUs; all operator measurements use tensor parallelism with $TP = 8$. The associative-recall study includes FullAttn (Vaswani et al., 2017), SWA (Beltagy et al., 2020), Seer (Gao et al., 2024), InfLLMv2 (Zhao et al., 2025b), NSA (Yuan et al., 2025), MoBA (Lu et al., 2025), DSA (DeepSeek-AI et al., 2025), and CoWA to cover dense, local, structured, and dynamic sparse designs. The latency-and-memory and scaling-law comparisons focus on FullAttn, MoBA, DSA, and CoWA, which provide the dense reference and representative end-to-end trainable sparse mechanisms. Complete configurations, per-query token budgets, training schedules, and evaluation protocols are provided in Appendix A. Associative recall matches visible-token budgets, whereas the operator and scaling comparisons approximately match selected-attention FLOPs among the sparse methods.

Collective Coverage with Matched Per-Head Window Widths. Does CoWA benefit from its window size, or from assigning different parts of the history to different heads? To distinguish these factors, we match the per-head window widths and vary the number of distinct long-range windows assigned across eight KV heads. The 1-, 2-, 4-, and 8-window layouts collectively cover 12.5%,

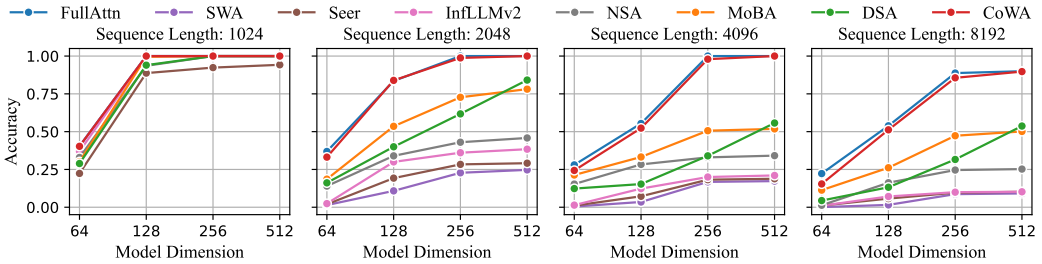

Figure 2: **Associative Recall**. Accuracy with 256 key-value pairs across sequence lengths and model dimensions. At sequence lengths 1K, 2K, 4K, and 8K, all sparse methods use matched per-query token budgets of 1,024, 1,152, 1,408, and 1,920 tokens, respectively. We do not add random padding or additional noise tokens. CoWA closely tracks FullAttn as the context grows, whereas the other sparse patterns lose a substantial fraction of the associations.

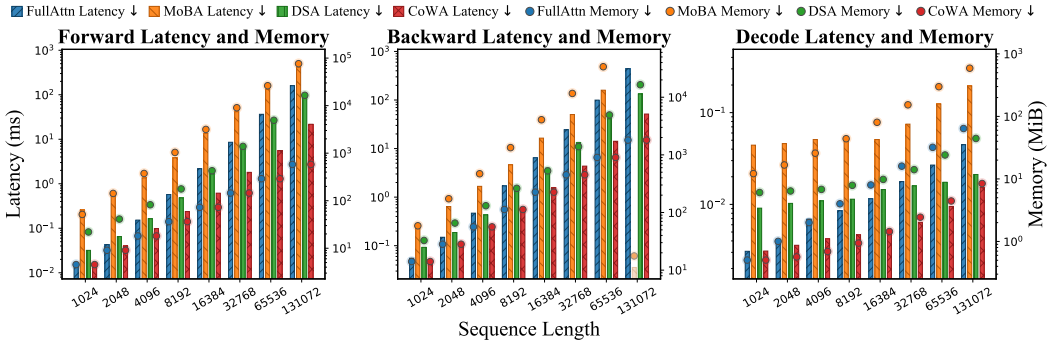

Figure 3: **Operator Latency and Memory**. End-to-end attention-operator latency (bars, left axes) and maximum per-rank peak allocation (markers, right axes) for the training forward and backward passes and decoding with tensor parallelism $TP = 8$ on 8 H100 GPUs. CoWA has the lowest latency and decoding-operator peak allocation among the tested methods at long sequence lengths.

25%, 50%, and 100% of the long-range history, respectively; Appendix A.1 details their construction. Table 1 shows that 8K recall increases monotonically from 21.32% to 32.92%, 52.32%, and 89.73% as duplicated windows are replaced by complementary ones. With full collective coverage, CoWA nearly matches FullAttn at 89.97%, while the near-only SWA reference reaches 5.34%. These results support the benefit of distributing complementary long-range access across KV heads.

Controlled Associative Recall. We next use a controlled associative-recall (Arora et al., 2024) task to test whether each attention mechanism can recover arbitrary key-value bindings from its accessible context. The randomized associations provide no semantic shortcut to the answer. Matching the model components outside attention helps isolate how each attention mechanism affects retrieval. Following the setup in Appendix A.1, we train on 256 key-value pairs, vary the sequence length from 1,024 to 8,192 and d_{model} from 64 to 512, and increase the matched per-query token budget from 1,024 to 1,920 tokens as the sequence grows. Figure 2 shows that, at larger model dimensions, all methods approach FullAttn when the budget covers the complete 1,024-token sequence, while their behavior separates at longer sequences. At sequence length 8,192 and $d_{model} = 512$, CoWA reaches 89.73% accuracy, essentially matching FullAttn at 89.97%; DSA and MoBA reach 53.71% and 50.12%, NSA reaches 25.23%, and the remaining methods stay near 10%. Several baselines attain higher accuracy at larger model dimensions, but increasing model dimension does not close the retrieval gap over the tested range.

Operator Latency and Memory. Having established the retrieval behavior of the attention mechanisms, we next measure the systems cost of executing the dense reference and the three representative trainable sparse designs. The benchmark (Tillet et al., 2019) includes sparse-pattern construction and data movement in addition to selected attention: MoBA includes block pooling, routing, TopK, rearrangement, and merging; DSA includes its lightning indexer, quantization, TopK, and sparse MLA; and CoWA includes its fused range and masking logic. All measurements use the matched $TP = 8$ setup described in Appendix A.2. Figure 3 shows that CoWA's regular attention pattern translates into consistent gains as the context grows. At 128K tokens, CoWA reduces forward and backward latency during training by $7.4\times$ and $8.6\times$, and decoding latency during inference by

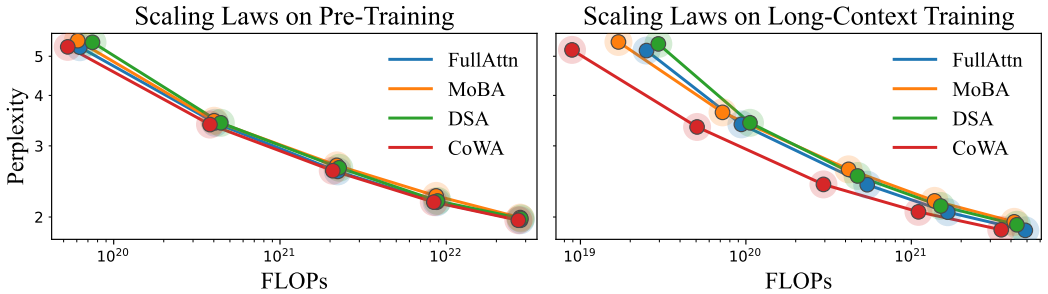

Figure 4: **Scaling Laws**. Perplexity as a function of training FLOPs for models from 0.6B to 14B parameters during 4K pre-training (left) and 32K long-context training (right). Each marker reports final perplexity; its outer ring denotes one standard deviation over the perplexity measurements collected during training. CoWA closely follows FullAttn in perplexity while requiring fewer FLOPs.

Table 2: **Model-Level Evaluation Summary**. Average knowledge, reasoning, and RULER scores at 14B and 32B. RULER is evaluated at the native 32K context length and under YaRN extrapolation to 128K. Detailed per-task results and uncertainty estimates are reported in Appendix B.

Model Scale	Method	Knowledge	Reasoning	RULER 32K	RULER 128K
14B	FullAttn	72.32	64.46	**89.42**	65.84
14B	MoBA	70.58	59.71	87.17	61.52
14B	DSA	70.23	63.28	87.67	64.05
14B	CoWA (ours)	**72.70**	**64.87**	89.13	**66.60**
32B	FullAttn	75.62	**75.67**	**92.70**	**82.03**
32B	CoWA (ours)	**76.07**	75.53	92.58	81.78

3.0× relative to FullAttn. Its forward and backward peak allocations match those of the FullAttn implementation, while its 8.4 MiB decoding-operator peak allocation is 7.6× smaller than FullAttn's and substantially smaller than MoBA's and DSA's. MoBA's 128K backward pass exceeds the per-rank memory limit. MoBA and DSA incur the routing and indexing costs described above; CoWA derives its visible blocks directly from sequence position and global KV-head index, avoiding these auxiliary structures.

Scaling Across Model Sizes. We next test the quality-compute trade-off during language-model training across model sizes (Kaplan et al., 2020; Xiong et al., 2023). We study five model scales from 0.6B to 14B under matched pre-training and long-context training configurations. Figure 4 reports perplexity against total training FLOPs, including the sequence-dependent attention and sparse-selection work. Across both stages, CoWA closely tracks FullAttn in perplexity while requiring less computation, giving the best overall perplexity-FLOPs trade-off among the sparse variants. At 14B, CoWA matches FullAttn's pre-training perplexity and differs by less than 0.01 after long-context training, while reducing total training FLOPs by 3.1% and 28.5%, respectively. The FLOP reduction is largest in the 32K stage, where the cost of dense attention grows with sequence length, while CoWA uses a per-QO-head token budget of 4,992 without introducing router or indexer FLOPs. Detailed model architectures, token budgets, FLOP accounting, and optimization settings are provided in Appendix A.3.

Model-Level Evaluation. We finally test whether the preceding trends transfer to complete language models. We benchmark all four 14B attention variants obtained from the scaling-law study and extend the FullAttn-CoWA comparison to the 32B models obtained through the separate continued-training experiment; the latter are included only in this evaluation and not in the scaling-law analysis. Table 2 shows that CoWA remains comparable to FullAttn in average knowledge, reasoning, native 32K retrieval, and YaRN-extrapolated 128K retrieval at both scales. At 14B, CoWA scores 72.70 and 64.87 on knowledge and reasoning versus 72.32 and 64.46 for FullAttn. At 32B, the corresponding averages are 76.07 and 75.53 versus 75.62 and 75.67. CoWA remains within 0.3 points of FullAttn at native 32K for both model scales. Under YaRN (Peng et al., 2026) extrapolation to 128K, CoWA reaches 66.60 versus 65.84 for FullAttn at 14B and 81.78 versus 82.03 at 32B. Bold and underlining mark the best and second-best results, respectively, within each column and model scale. Detailed per-task results and uncertainty estimates are reported in Appendix B. The 14B results connect lower training FLOPs to FullAttn-level average benchmark performance, while the 32B comparison shows that comparable quality can also be retained after continued training.

4 Related Work

CoWA defines sparse access directly from position during training and inference, replacing duplicated access to the causal history with complementary long-range windows across KV heads; we relate this design to sparse attention, head specialization and routing, hardware-aware sparse execution, and KV-cache management and distributed inference.

Sparse attention. Fixed sparse attention uses local windows, structured blocks, or global tokens (Child et al., 2019; Beltagy et al., 2020; Zaheer et al., 2020). Sliding-window attention allows regular kernel execution, but restricts every head to the same finite horizon (Fu et al., 2025). Dynamic methods select tokens or blocks using retrieval scores, importance estimates, or learned routers (Tang et al., 2024; Li et al., 2024; Zhang et al., 2023; Xiao et al., 2024a; Desai et al., 2024; Zhang et al., 2025; Gao et al., 2024; Yuan et al., 2025; Lu et al., 2025; Lai et al., 2025). CoWA combines shared near and sink windows with complementary long-range windows across KV heads, retaining full causal coverage through a position-defined rule without content-dependent selection.

Head specialization. Attention heads exhibit distinct local and long-range roles (Guo et al., 2024; Gu et al., 2024; Barbero et al., 2025; Xiao et al., 2024b; Sandoval-Segura et al., 2026). SPAttention distributes causal-distance regions across MHA heads while preserving aggregate causal connectivity (Zhao et al., 2025a).[1] We systematically evaluate collective coverage as a design principle for sparse attention: an attention layer retains access to the full causal history collectively across heads, without requiring every head to attend to every position. CoWA's complementary KV-head windows provide a simple instantiation of this principle under GQA. This allocation of visible keys differs from token-dependent gating of head outputs (Fu et al., 2026). Beyond the particular distance partition, we study this principle through controlled coverage ablations, evaluations of language-model quality and long-range retrieval, and benchmarks of tensor-parallel training and inference operators.

Hardware-aware sparse execution. FlashAttention reduces memory traffic through tiled execution and online softmax while preserving dense attention (Dao et al., 2022; Shah et al., 2024); FlexAttention and FlashInfer improve kernel programmability and serving efficiency (Dong et al., 2024; Ye et al., 2025). CoWA uses tiled block traversal but derives visible block ranges directly from position and global KV-head index, avoiding separate routing, indexing, and materialized selection state. Its position-defined sparsity is part of the trained attention architecture rather than an inference-only approximation.

KV-cache management and distributed inference. KV-cache systems reduce inference cost by pruning, compressing, retrieving, or offloading cached states (Zhang et al., 2023; Li et al., 2024; Tang et al., 2024; Xiao et al., 2024a; Liu et al., 2024; Huang et al., 2025). CoWA instead changes the trained attention architecture and aligns complementary windows with tensor-parallel KV-head sharding; cache compression and offloading remain orthogonal.

5 Conclusion

We introduced CoWindow Attention (CoWA), a structured architecture that distributes access to the causal history across KV heads using one position-defined window rule for training and inference. CoWA shares near and sink windows while assigning complementary long-range windows to different heads, making full causal coverage a collective property of the head ensemble. This design reduces duplicated long-range access without a learned router or indexer. Global KV-head indexing aligns the window assignment with tensor parallelism and per-rank KV-cache partitioning.

The window-matched ablation shows that complementary long-range windows improve recall compared with duplicated windows at the same per-head widths. The controlled associative recall further shows that CoWA closely tracks FullAttn as context length grows. The tensor-parallel operator benchmarks show lower attention latency and peak decoding-operator memory. The scaling-law experiments from 0.6B to 14B parameters show perplexity comparable to FullAttn with lower training FLOPs, and the model-level evaluations at 14B and 32B show comparable knowledge, reasoning, and long-context retrieval scores. These results indicate that distributing long-range access across KV heads can retain the evaluated capabilities of FullAttn while reducing attention computation.

[1] SPAttention's published exclusive bands differ from its released shared-local implementation `MEHC`, which does not provide the long-context GQA, decoding, or tensor-parallel execution studied here.

AI USE STATEMENT

We used generative AI tools to assist with drafting and language editing of the manuscript, including improving grammar, wording, paragraph flow, and the clarity of technical descriptions. The authors reviewed and revised the AI-assisted text and take full responsibility for the final manuscript, including its technical claims and experimental results.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints, 2023. URL https://arxiv.org/abs/2305.13245.

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. In *The International Conference on Learning Representations*, 2024.

Federico Barbero, 'Alvaro Arroyo, Xiangming Gu, Christos Perivolaropoulos, Michael M. Bronstein, Petar Velivckovi 'c, and Razvan Pascanu. Why do llms attend to the first token? *ArXiv*, abs/2504.02732, 2025.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Loubna Ben Allal, Anton Lozhkov, Guilherme Penedo, Thomas Wolf, and Leandro von Werra. Smollm-corpus, July 2024.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on Artificial Intelligence*, volume 34, 2020.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

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.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: fast and memory-efficient exact attention with io-awareness. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, 2022.

Google DeepMind. Gemini 2.5 pro, March 2025. URL https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025.

DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenhao Xu, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Erhang Li, Fangqi Zhou, Fangyun Lin, Fucong Dai, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Li, Haofen Liang, Haoran Wei, Haowei Zhang, Haowen Luo, Haozhe Ji, Honghui Ding, Hongxuan Tang, Huanqi Cao, Huazuo Gao, Hui Qu, Hui Zeng, Jialiang Huang, Jiashi Li, Jiaxin Xu, Jiewen Hu, Jingchang Chen, Jingting Xiang, Jingyang Yuan, Jingyuan Cheng, Jinhua Zhu, Jun Ran, Junguang Jiang, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Kexin Huang, Kexing Zhou, Kezhao Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Wang, Liang Zhao, Liangsheng Yin, Lihua Guo, Lingxiao Luo, Linwang Ma, Litong Wang, Liyue Zhang, M. S. Di, M. Y Xu, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Panpan Huang, Peixin Cong, Peiyi Wang, Qiancheng Wang, Qihao Zhu, Qingyang Li, Qinyu

Chen, Qiushi Du, Ruiling Xu, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runqiu Yin, Runxin Xu, Ruomeng Shen, Ruoyu Zhang, S. H. Liu, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaofei Cai, Shaoyuan Chen, Shengding Hu, Shengyu Liu, Shiqiang Hu, Shirong Ma, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, Songyang Zhou, Tao Ni, Tao Yun, Tian Pei, Tian Ye, Tianyuan Yue, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjie Pang, Wenjing Luo, Wenjun Gao, Wentao Zhang, Xi Gao, Xiangwen Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaokang Zhang, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xingyou Li, Xinyu Yang, Xinyuan Li, Xu Chen, Xuecheng Su, Xuehai Pan, Xuheng Lin, Xuwei Fu, Y. Q. Wang, Yang Zhang, Yanhong Xu, Yanru Ma, Yao Li, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Qian, Yi Yu, Yichao Zhang, Yifan Ding, Yifan Shi, Yiliang Xiong, Ying He, Ying Zhou, Yinmin Zhong, Yishi Piao, Yisong Wang, Yixiao Chen, Yixuan Tan, Yixuan Wei, Yiyang Ma, Yiyuan Liu, Yonglun Yang, Yongqiang Guo, Yongtong Wu, Yu Wu, Yuan Cheng, Yuan Ou, Yuanfan Xu, Yuduan Wang, Yue Gong, Yuhan Wu, Yuheng Zou, Yukun Li, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehua Zhao, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhixian Huang, Zhiyu Wu, Zhuoshu Li, Zhuping Zhang, Zian Xu, Zihao Wang, Zihui Gu, Zijia Zhu, Zilin Li, Zipeng Zhang, Ziwei Xie, Ziyi Gao, Zizheng Pan, Zongqing Yao, Bei Feng, Hui Li, J. L. Cai, Jiaqi Ni, Lei Xu, Meng Li, Ning Tian, R. J. Chen, R. L. Jin, S. S. Li, Shuang Zhou, Tianyu Sun, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xinnan Song, Xinyi Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, Dongjie Ji, Jian Liang, Jianzhong Guo, Jin Chen, Leyi Xia, Miaojun Wang, Mingming Li, Peng Zhang, Ruyi Chen, Shangmian Sun, Shaoqing Wu, Shengfeng Ye, T. Wang, W. L. Xiao, Wei An, Xianzu Wang, Xiaowen Sun, Xiaoxiang Wang, Ying Tang, Yukun Zha, Zekai Zhang, Zhe Ju, Zhen Zhang, and Zihua Qu. Deepseek-v3.2: Pushing the frontier of open large language models, 2025. URL https://arxiv.org/abs/2512.02556.

Aditya Desai, Shuo Yang, Alejandro Cuadron, Ana Klimovic, Matei Zaharia, Joseph E Gonzalez, and Ion Stoica. Hashattention: Semantic sparsity for faster inference. *arXiv preprint arXiv:2412.14468*, 2024.

Juechu Dong, Boyuan Feng, Driss Guessous, Yanbo Liang, and Horace He. Flex attention: A programming model for generating optimized attention kernels, 2024. URL https://arxiv.org/abs/2412.05496.

Zichuan Fu, Wentao Song, Yejing Wang, Xian Wu, Yefeng Zheng, Yingying Zhang, Derong Xu, Xuetao Wei, Tong Xu, and Xiangyu Zhao. Sliding window attention training for efficient large language models, 2025. URL https://arxiv.org/abs/2502.18845.

Zizhuo Fu, Wenxuan Zeng, Runsheng Wang, and Meng Li. Attention sink forges native moe in attention layers: Sink-aware training to address head collapse, 2026. URL https://arxiv.org/abs/2602.01203.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, Jason Phang, Laria Reynolds, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, September 2021. URL https://doi.org/10.5281/zenodo.5371628.

Yizhao Gao, Zhichen Zeng, Dayou Du, Shijie Cao, Peiyuan Zhou, Jiaxing Qi, Junjie Lai, Hayden Kwok-Hay So, Ting Cao, Fan Yang, et al. Seerattention: Learning intrinsic sparse attention in your llms. *arXiv preprint arXiv:2410.13276*, 2024.

Xiangming Gu, Tianyu Pang, Chao Du, Qian Liu, Fengzhuo Zhang, Cunxiao Du, Ye Wang, and Min Lin. When attention sink emerges in language models: An empirical view. *ArXiv*, abs/2410.10781, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Tianyu Guo, Druv Pai, Yu Bai, Jiantao Jiao, Michael I. Jordan, and Song Mei. Active-dormant attention heads: Mechanistically demystifying extreme-token phenomena in llms. *ArXiv*, abs/2410.13835, 2024.

Alex Hägele, Elie Bakouch, Atli Kosson, Leandro Von Werra, Martin Jaggi, et al. Scaling laws and compute-optimal training beyond fixed training durations. *Advances in Neural Information Processing Systems*, 37:76232–76264, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021a.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021b. URL https://arxiv.org/abs/2103.03874.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. An empirical analysis of compute-optimal large language model training. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:30016–30030, 2022.

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? *arXiv preprint arXiv:2404.06654*, 2024.

Yuxiang Huang, Chaojun Xiao, Xu Han, and Zhiyuan Liu. Nosa: Native and offloadable sparse attention, 2025. URL https://arxiv.org/abs/2510.13602.

HuggingFace. Open r1: A fully open reproduction of deepseek-r1, 2025. URL https://github.com/huggingface/open-r1.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Xunhao Lai, Jianqiao Lu, Yao Luo, Yiyuan Ma, and Xun Zhou. Flexprefill: A context-aware sparse attention mechanism for efficient long-sequence inference, 2025. URL https://arxiv.org/abs/2502.20766.

Houyi Li, Wenzhen Zheng, Qiufeng Wang, Hanshan Zhang, Zili Wang, Shijie Xuyang, Yuantao Fan, Shuigeng Zhou, Xiangyu Zhang, and Daxin Jiang. Predictable scale: Part i – optimal hyperparameter scaling law in large language model pretraining, 2025. URL https://arxiv.org/abs/2503.04715.

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. *Advances in Neural Information Processing Systems*, 37:22947–22970, 2024.

Guangda Liu, Chengwei Li, Jieru Zhao, Chenqi Zhang, and Minyi Guo. Clusterkv: Manipulating llm kv cache in semantic space for recallable compression. *arXiv preprint arXiv:2412.03213*, 2024.

Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *ArXiv*, abs/1711.05101, 2017. URL https://api.semanticscholar.org/CorpusID:3312944.

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, Zhiqi Huang, Huan Yuan, Suting Xu, Xinran Xu, Guokun Lai, Yanru Chen, Huabin Zheng, Junjie Yan, Jianlin Su, Yuxin Wu, Yutao Zhang, Zhilin Yang, Xinyu Zhou, Mingxing Zhang, and Jiezhong Qiu. Moba: Mixture of block attention for long-context llms. *arXiv preprint arXiv:2502.13189*, 2025.

Chenyang Lyu, Minghao Wu, and Alham Fikri Aji. Beyond probabilities: Unveiling the misalignment in evaluating large language models, 2024. URL https://arxiv.org/abs/2402.13887.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.

Maxim Milakov and Natalia Gimelshein. Online normalizer calculation for softmax, 2018. URL https://arxiv.org/abs/1805.02867.

Meta NVIDIA. Pytorch container image. https://catalog.ngc.nvidia.com/orgs/nvidia/containers/pytorch, 2022.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pp. 1–22, 2023.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. Yarn: Efficient context window extension of large language models, 2026. URL https://arxiv.org/abs/2309.00071.

Yanlin Qi, Xinhang Chen, Huiqiang Jiang, Qitong Wang, Botao Peng, and Themis Palpanas. Pariskv: Fast and drift-robust kv-cache retrieval for long-context llms, 2026. URL https://arxiv.org/abs/2602.07721.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof qa benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial Winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Pedro Sandoval-Segura, Xijun Wang, Ashwinee Panda, Micah Goldblum, Ronen Basri, Tom Goldstein, and David Jacobs. Identifying and evaluating inactive heads in pretrained llms. In *International Conference on Learning Representations*, 2026.

Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. Flashattention-3: Fast and accurate attention with asynchrony and low-precision, 2024. URL https://arxiv.org/abs/2407.08608.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, 2023.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context llm inference. *arXiv preprint arXiv:2406.10774*, 2024.

Qwen Team. Qwen3, April 2025. URL https://qwenlm.github.io/blog/qwen3.

Philippe Tillet, H. T. Kung, and David Cox. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pp. 10–19. Association for Computing Machinery, 2019.

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*, 2017.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark, 2024. URL https://arxiv.org/abs/2406.01574.

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. 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, Online, October 2020. Association for Computational Linguistics.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. Infllm: Training-free long-context extrapolation for llms with an efficient context memory. *arXiv preprint arXiv:2402.04617*, 2024a.

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. Duoattention: Efficient long-context llm inference with retrieval and streaming heads. *ArXiv*, abs/2410.10819, 2024b.

Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, et al. Effective long-context scaling of foundation models. *arXiv preprint arXiv:2309.16039*, 2023.

Zihao Ye, Lequn Chen, Ruihang Lai, Wuwei Lin, Yineng Zhang, Stephanie Wang, Tianqi Chen, Baris Kasikci, Vinod Grover, Arvind Krishnamurthy, and Luis Ceze. Flashinfer: Efficient and customizable attention engine for llm inference serving, 2025. URL https://arxiv.org/abs/2501.01005.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, YX Wei, Lean Wang, Zhiping Xiao, et al. Native sparse attention: Hardware-aligned and natively trainable sparse attention. *arXiv preprint arXiv:2502.11089*, 2025.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020.

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.

Hanzhi Zhang, Heng Fan, Kewei Sha, Yan Huang, and Yunhe Feng. Dam: Dynamic attention mask for long-context large language model inference acceleration, 2025. URL https://arxiv.org/abs/2506.11104.

Kechi Zhang, Jia Li, Ge Li, Xianjie Shi, and Zhi Jin. Codeagent: Enhancing code generation with tool-integrated agent systems for real-world repo-level coding challenges. *arXiv preprint arXiv:2401.07339*, 2024.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710, 2023.

Mingkuan Zhao, Wentao Hu, Jiayin Wang, Xin Lai, Tianchen Huang, Yuheng Min, Rui Yan, and Xiaoyan Zhu. Making every head count: Sparse attention without the speed-performance trade-off, 2025a. URL https://arxiv.org/abs/2511.09596.

Weilin Zhao, Zihan Zhou, Zhou Su, Chaojun Xiao, Yuxuan Li, Yanghao Li, Yudi Zhang, Weilun Zhao, Zhen Li, Yuxiang Huang, Ao Sun, Xu Han, and Zhiyuan Liu. Infllm-v2: Dense-sparse switchable attention for seamless short-to-long adaptation, 2025b. URL https://arxiv.org/abs/2509.24663.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. Agieval: A human-centric benchmark for evaluating foundation models, 2023. URL https://arxiv.org/abs/2304.06364.

A EXPERIMENT SETTINGS

Within each experiment, attention variants use matched model scales, depth, hidden size, data, and optimization settings; attention-specific head configurations and sparse hyperparameters follow each method. Our software stack uses NVIDIA PyTorch container images (NVIDIA, 2022) and the Transformers framework (Wolf et al., 2020). The scaling-law (Kaplan et al., 2020; Xiong et al., 2023) and continued-training experiments use Megatron-LM (Shoeybi et al., 2019) for distributed training on 128 NVIDIA H100 GPUs. We use the EleutherAI LM Evaluation Harness (Gao et al., 2021) for benchmark evaluation on 8 NVIDIA H100 GPUs. Operator benchmarking uses 8 H100 GPUs with tensor parallelism $TP = 8$.

A.1 ASSOCIATIVE RECALL SETUP

Following prior work (Arora et al., 2024), we evaluate associative recall with 256 key-value pairs. We consider sequence lengths of 1,024, 2,048, 4,096, and 8,192, and use the standard model dimensions $d_{model} \in \{64, 128, 256, 512\}$. The repeated key-value content is placed contiguously at the beginning of each sequence, occupying the first $2n_{\text{kv}}n_{\text{pass}}$ tokens, where $n_{\text{kv}} = 256$ and n_{pass} is the number of passes. We then sample n_{kv} query positions without replacement from the remaining positions according to $p(t) \propto t^{a-1}$, where t is the position within this suffix. We use the default $a = 0.01$, so queries occur more frequently near the beginning of the suffix and become progressively sparser toward the end. We do not insert random padding or additional noise tokens. This setup tests recall without adding a separate task of filtering random padding or noise.

The dataset contains 250K training examples and 1K test examples, and all models are trained for 100 epochs. At the four sequence lengths, every sparse attention variant uses matched per-query token budgets of 1,024, 1,152, 1,408, and 1,920 tokens, respectively. For CoWA, we use $w_{\text{sink}} = 1$ and $w_{\text{near}} = 1{,}023$, giving a shared width of 1,024 tokens; the complementary long-range windows produce the stated effective per-QO-head token budgets after overlap and sequence-boundary clipping.

Collective-coverage layouts. We index the eight equal-width long-range regions from the current query toward the beginning of the sequence: W_1 is adjacent to the near window and W_8 is adjacent to the sink window. Reading from the earliest token to the current query, the union of the CoWA heads is therefore $\text{sink} \mid W_8 \mid W_7 \mid \cdots \mid W_2 \mid W_1 \mid \text{near}$. All CoWA heads share the sink and near windows, and each KV head receives one distinct W_j. In the collective-coverage ablation, SWA receives no sink or long-range window and attends only through its near window; it is included as a local reference rather than a work-matched variant. For an m-window layout, we repeat the m windows closest to the near window, $W_1, \ldots, W_m$, across the eight KV heads. Thus, 1, 2, or 4 distinct long-range windows are each repeated across 8, 4, or 2 KV heads, respectively. These layouts and CoWA use matched per-head window widths while increasing collective coverage.

A.2 OPERATOR ACCELERATION SETUP

We separate the cost of sparse attention into computation over selected tokens and the work required to determine the sparse pattern. Let n denote sequence length, h the number of QO heads, d_{qk} and d_v the query/key and value head dimensions, B the MoBA block size, and w the per-query token budget. Thus, w denotes the number of selected tokens after expanding MoBA blocks, the number of top-ranked tokens selected by DSA, and the per-QO-head token budget for CoWA. For DSA, h_I and d_I denote the number and dimension of lightning-indexer heads. Table 3 reports the resulting asymptotic costs. Projection layers and other sequence-linear Transformer operations are shared model costs and are omitted from this table.

FullAttn computes QK and PV over all token pairs. MoBA (Lu et al., 2025) reduces the main attention to selected blocks, but first mean-pools block keys and evaluates every query against the n/B block representatives. DSA (DeepSeek-AI et al., 2025) performs token-level sparse MLA after a lightning indexer scores all query-key pairs and generates TopK indices. CoWA requires no separate routing or indexing stage; its fused range calculation directly restricts QK and PV to the retained tokens.

Consequently, methods with the same selected-token budget can have substantially different end-to-end costs once routing and indexing are included.

Table 3: **Comparison of Different Attention Variants**. The sequence-dependent attention and sparse-selection costs. The auxiliary state column describes routing or indexing state beyond the QKV tensors and attention outputs.

MECHANISM	SELECTED ATTENTION	SELECTION OVERHEAD	AUXILIARY STATE
FullAttn	$O\big(n^2h(d_{qk}+d_v)\big)$	-	-
MoBA	$O(nwh(d_{qk}+d_v))$	$O\big(n^2hd_{qk}/B\big)$	$O\big(n^2h/B\big)$
DSA	$O(nwh(d_{qk}+d_v))$	$O\big(n^2h_Id_I\big)$	$O\big(n^2h_I+nw\big)$
CoWA	$O(nwh(d_{qk}+d_v))$	$O(1)$ per QO block	$O(H_k)$

We benchmark (Tillet et al., 2019) causal attention at sequence lengths from 1,024 to 131,072 on 8 NVIDIA H100 GPUs (SM90) with tensor parallelism $TP = 8$. All methods use batch size one and the same tensor-parallel configuration. Unless a configuration runs out of memory, latency is averaged over 100 iterations after 25 warm-up iterations. We report the synchronized wall-clock time of the tensor-parallel group, and peak memory is the maximum per-rank allocation across the eight ranks. Table 4 summarizes the operator shapes, precision, and sparse-selection mechanism used by each method, while Table 5 reports the exact retained-token budgets passed at every sequence length.

The **Per-Selected-Token FLOPs** (PST FLOPs) column reports the forward QK and PV FLOPs per query contributed by one selected key token, summed over all QO heads. Counting one multiply-accumulate as two FLOPs, it is $2h(d_{qk}+d_v)$. This quantity excludes router, indexer, TopK, and other sparse-pattern construction costs; it is used only to approximately match the forward selected-attention workload. MoBA budgets are aligned to its 128-token blocks, while DSA uses approximately one quarter of the CoWA budget because its PST FLOPs are approximately four times as large. Total training FLOPs separately count QK recomputation and the value-side backward operations. FullAttn always attends to the complete causal context.

The reported measurements of the training forward and backward passes and inference-time decoding are operator-level end-to-end timings rather than isolated sparse-attention kernel timings. In particular, MoBA includes block representative construction, router logits, TopK, index construction, data rearrangement, and output merging. DSA includes BF16 indexer inputs, FP8 quantization, indexer logits, TopK/indices, and sparse MLA; decoding quantizes and appends only the newly generated Q/KV token before index selection and sparse attention. CoWA includes its fused window-range and masking logic. Model-level QKV/output projections, MLP layers, communication, and optimizer updates are outside the scope of this module benchmark. Peak memory is measured over the same end-to-end operator path and does not represent persistent serving-time KV-cache size.

Table 4: **Latency and Memory Benchmark Configurations**. The operator shapes and sparse-selection mechanisms used for the running-time curves. All measurements use tensor parallelism $TP = 8$ on 8 H100 GPUs.

ALGO	warmups	runs	batch	n_h	$n_{h_{kv}}$	d_{qk}	d_v	selector	precision
FullAttn[1]	25	100	1	64	8	128	128	-	BF16
MoBA[2]	25	100	1	64	64	128	128	$B = 128$	BF16
DSA[3,4]	25	100	1	64	1	576	512	FP8 indexer	BF16/FP8
CoWA	25	100	1	64	8	128	128	fused window	BF16

[1]The FullAttn implementation is available at `github.com/Dao-AILab/flash-attention`.

[2]The MoBA implementation is available at `https://github.com/MoonshotAI/MoBA`.

[3]The DSA indexer implementation is available at `github.com/deepseek-ai/DeepGEMM`.

[4]The DSA implementation is available at `github.com/Dao-AILab/flash-attention/flash_attn/cute`.

Table 5: **Per-Query Token Budgets**. The token budgets used at every benchmark sequence length.

ALGO	PST FLOPS	1K	2K	4K	8K	16K	32K	64K	128K
FullAttn	32768	1024	2048	4096	8192	16384	32768	65536	131072
MoBA	32768	1024	1024	1536	2048	3072	5120	9216	17408
DSA	139264	256	288	352	480	736	1248	2272	4320
CoWA	32768	1024	1152	1408	1920	2944	4992	9088	17280

Table 6: **Latency and Memory Benchmark Results**. End-to-end latency (ms) and peak memory (MiB) for the training forward and backward passes and inference-time decoding at every sequence length. Latency is the synchronized tensor-parallel group latency, and memory is the maximum per-rank peak allocation.

ALGO	SeqLen	Fwd Lat.	Bwd Lat.	Dec Lat.	Fwd Mem.	Bwd Mem.	Dec Mem.
FullAttn	1,024	0.015	0.055	0.003	4.531	14.094	0.504
FullAttn	2,048	0.043	0.149	0.004	9.063	28.188	1.004
FullAttn	4,096	0.153	0.468	0.007	18.125	56.375	2.004
FullAttn	8,192	0.572	1.699	0.009	36.250	112.750	4.004
FullAttn	16,384	2.179	6.455	0.012	72.500	225.500	8.004
FullAttn	32,768	8.570	24.449	0.018	145.000	451.000	16.004
FullAttn	65,536	36.431	99.664	0.027	290.000	902.000	32.004
FullAttn	131,072	159.987	441.230	0.045	580.000	1,804.001	64.004
MoBA	1,024	0.259	0.231	0.044	51.425	58.825	12.162
MoBA	2,048	0.563	0.639	0.046	142.287	173.275	16.712
MoBA	4,096	1.348	1.668	0.051	373.875	470.188	25.800
MoBA	8,192	3.833	4.638	0.051	1,035.975	1,335.475	43.987
MoBA	16,384	13.577	16.429	0.051	3,156.900	4,069.559	80.438
MoBA	32,768	41.543	50.268	0.075	9,029.144	11,639.466	153.125
MoBA	65,536	131.851	159.542	0.124	26,236.976	33,822.076	298.637
MoBA	131,072	418.473	OOM	0.196	76,239.664	OOM	589.650
DSA	1,024	0.032	0.093	0.009	21.988	32.725	6.062
DSA	2,048	0.065	0.187	0.010	41.100	65.562	6.412
DSA	4,096	0.164	0.431	0.011	82.200	131.125	6.800
DSA	8,192	0.481	1.324	0.011	176.887	262.750	7.900
DSA	16,384	1.668	3.955	0.015	427.100	527.500	9.825
DSA	32,768	6.231	13.199	0.016	1,388.225	1,388.225	14.075
DSA	65,536	24.616	40.914	0.017	4,924.562	4,924.562	24.150
DSA	131,072	93.671	133.231	0.021	16,479.996	16,479.996	44.388
CoWA	1,024	0.015	0.041	0.003	4.531	14.094	0.504
CoWA	2,048	0.040	0.106	0.003	9.063	28.188	0.566
CoWA	4,096	0.098	0.253	0.004	18.125	56.375	0.691
CoWA	8,192	0.237	0.600	0.004	36.250	112.750	0.941
CoWA	16,384	0.616	1.564	0.005	72.500	225.500	1.441
CoWA	32,768	1.807	4.358	0.006	145.000	451.000	2.441
CoWA	65,536	5.573	14.150	0.009	290.000	902.000	4.441
CoWA	131,072	21.758	51.509	0.015	580.000	1,804.001	8.441

A.3 SCALING LAWS SETUP

The scaling-law (Kaplan et al., 2020; Xiong et al., 2023) experiments use SmolLMCorpus (Ben Allal et al., 2024), the Qwen3 tokenizer (Team, 2025), the AdamW optimizer (Loshchilov & Hutter, 2017), the WSD learning-rate schedule (Hägele et al., 2024), and the compute-optimal scaling principles (Li et al., 2025; Hoffmann et al., 2022). We use a fixed random seed of 42 for all scaling-law runs. We summarize the meaning of the columns in Table 7 and clarify which hyperparameters are used by each attention variant.

- **Params**: target model scale. Parameter counts are matched approximately because the attention projections differ across variants.
- **PT/LCT Tok**: total numbers of tokens used in pre-training (PT) and long-context training (LCT), respectively. Each table entry reports the two values in the order `PT, LCT`.
- **Batch**: tokens per optimization step.
- **PT/LCT LR**: learning rates used during PT and LCT, respectively. Each table entry reports the two values in the order `PT, LCT`.
- n_{layers}: number of Transformer layers.
- d_{model}: model hidden size.
- n_h: number of attention heads.
- $n_{h_{kv}}$: number of KV heads.
- d_{qk} and d_v: per-head dimensions of the query/key and value representations, respectively.
- **Per-Selected-Token FLOPs** (PST FLOPs): forward QK and PV FLOPs per query contributed by each selected key token, summed over all QO heads, as defined in Appendix A.2.
- **PT Bgt** and **LCT Bgt**: the per-query token budgets used during pre-training and long-context training, respectively.

For models from 0.6B to 14B, training proceeds in two stages. PT uses a sequence length of 4,096 and a WSD learning-rate schedule (Hägele et al., 2024) whose peak learning rate is reported in the PT entry of Table 7. The decay phase ends at the corresponding LCT learning rate, which is $0.1\times$ the PT peak learning rate. LCT initializes from the final PT checkpoint, increases the sequence length to 32,768, and uses the reported constant LCT learning rate.

- **FullAttn**: standard causal scaled dot-product attention over the complete pre-training or long-context sequence.
- **MoBA**: block-sparse attention with one KV head per QO head and retained-token budgets of 1,536 during pre-training and 5,120 during long-context training.
- **DSA**: sparse latent attention with one KV head, $d_{qk} = 576$, and $d_v = 512$. Since each selected token requires approximately four times the forward QK and PV FLOPs of MoBA or CoWA, we use budgets of 352 and 1,248.
- **CoWA**: grouped-query sparse attention with eight KV heads and per-QO-head token budgets of 1,408 during pre-training and 4,992 during long-context training.

We retain each method's native head configuration: FullAttn and CoWA use eight KV heads, MoBA uses one KV head per QO head, and DSA uses a single latent KV head. For the sparse variants, we approximately match forward selected-attention FLOPs rather than head counts or raw token budgets. The token budgets differ because of each method's per-token cost and block or token-selection granularity. This matching criterion excludes method-specific routing, indexing, and top-k overheads and does not require identical end-to-end training FLOPs. FullAttn attends to the complete context and serves as the dense reference.

The scaling-law plots use cumulative training FLOPs reported by Megatron-LM and therefore cover the complete model computation. To expose the attention-specific contribution to this statistic, we additionally measure the exact number of legal attention connections executed under each training configuration rather than assigning the nominal final-query budget to every query. Let $\bar{s}$ denote the resulting mean number of selected keys per query and QO head. Attention backward recomputes QK scores, so the sequence-dependent core-attention cost per token and layer is $8\bar{s}hd_{qk} + 6\bar{s}hd_v$, counting one multiply-accumulate as two FLOPs. FullAttn has $\bar{s} = (n+1)/2$ over the causal triangle. For CoWA, averaging the exact support sizes over all query positions and QO heads gives $\bar{s} = 1{,}040.17$ during 4K pre-training and $\bar{s} = 2{,}930.08$ during 32K long-context training. MoBA and DSA use their causally clipped selected supports, with their routing and indexing costs included in the reported totals.

Table 7: **Self-Attention Variants Scaling Laws Configurations**. The model and hyperparameter configurations used in our self-attention variants scaling laws experiments.

ALGOS	PARAMS	PT/LCT TOK	BATCH	PT/LCT LR	n_{layers}	d_{model}	n_h	$n_{h_{kv}}$	d_{qk}	d_v	PST FLOPS	PT BGT	LCT BGT
FullAttn	≈ 0.6B	12B,1.5B	0.256M	1e-3,1e-4	28	1024	16	8	128	128	8192	4096	32768
MoBA	≈ 0.6B	12B,1.5B	0.256M	1e-3,1e-4	28	1024	16	16	128	128	8192	1536	5120
DSA	≈ 0.6B	12B,1.5B	0.256M	1e-3,1e-4	28	1024	16	1	576	512	34816	352	1248
CoWA	≈ 0.6B	12B,1.5B	0.256M	1e-3,1e-4	28	1024	16	8	128	128	8192	1408	4992
FullAttn	≈ 1.7B	34B,4B	0.512M	8e-4,8e-5	28	2048	16	8	128	128	8192	4096	32768
MoBA	≈ 1.7B	34B,4B	0.512M	8e-4,8e-5	28	2048	16	16	128	128	8192	1536	5120
DSA	≈ 1.7B	34B,4B	0.512M	8e-4,8e-5	28	2048	16	1	576	512	34816	352	1248
CoWA	≈ 1.7B	34B,4B	0.512M	8e-4,8e-5	28	2048	16	8	128	128	8192	1408	4992
FullAttn	≈ 4B	80B,10B	1M	6e-4,6e-5	32	2560	32	8	128	128	16384	4096	32768
MoBA	≈ 4B	80B,10B	1M	6e-4,6e-5	32	2560	32	32	128	128	16384	1536	5120
DSA	≈ 4B	80B,10B	1M	6e-4,6e-5	32	2560	32	1	576	512	69632	352	1248
CoWA	≈ 4B	80B,10B	1M	6e-4,6e-5	32	2560	32	8	128	128	16384	1408	4992
FullAttn	≈ 8B	164B,20B	1.6M	4e-4,4e-5	36	4096	32	8	128	128	16384	4096	32768
MoBA	≈ 8B	164B,20B	1.6M	4e-4,4e-5	36	4096	32	32	128	128	16384	1536	5120
DSA	≈ 8B	164B,20B	1.6M	4e-4,4e-5	36	4096	32	1	576	512	69632	352	1248
CoWA	≈ 8B	164B,20B	1.6M	4e-4,4e-5	36	4096	32	8	128	128	16384	1408	4992
FullAttn	≈ 14B	296B,36B	2M	3e-4,3e-5	40	5120	40	8	128	128	20480	4096	32768
MoBA	≈ 14B	296B,36B	2M	3e-4,3e-5	40	5120	40	40	128	128	20480	1536	5120
DSA	≈ 14B	296B,36B	2M	3e-4,3e-5	40	5120	40	1	576	512	87040	352	1248
CoWA	≈ 14B	296B,36B	2M	3e-4,3e-5	40	5120	40	8	128	128	20480	1408	4992

A.4 SPARSE ADAPTATION VIA CONTINUED TRAINING

We study sparse adaptation at 32B through long-context continued training, using FullAttn as the dense reference. Both models are trained at a sequence length of 32,768 for 64B tokens with a batch size of 4M tokens and a constant learning rate of 1×10^{-5}. FullAttn uses the full 32,768-token causal context, and CoWA uses a per-QO-head token budget of 4,992. Both configurations use 64 layers, a hidden size of 5,120, 64 QO heads, 8 KV heads, and per-head dimensions $d_{qk} = d_v = 128$.

A.5 BENCHMARK EVALUATION

We evaluate two groups of models. At 14B, we compare all four attention variants obtained from the scaling-law study in Appendix A.3. At 32B, we compare the FullAttn and CoWA models obtained from the continued-training setup in Appendix A.4. For each task, we report the mean and standard deviation over five evaluation runs with seeds 0, 42, 233, 666, and 1234. The standard benchmark suite covers knowledge, reasoning, and retrieval through MMLU (Hendrycks et al., 2021a; Lyu et al., 2024), MMLU-Pro (Wang et al., 2024), BBH (Suzgun et al., 2023), HellaSwag (Zellers et al., 2019), OBQA (Mihaylov et al., 2018), WinoGrande (Sakaguchi et al., 2021), PIQA (Bisk et al., 2020), GSM8K (Cobbe et al., 2021), Hendrycks-Math (Hendrycks et al., 2021b), ARC-C (Clark et al., 2018), AGIEval (Zhong et al., 2023), GPQA-Diamond (Rein et al., 2023), and RULER (Hsieh et al., 2024). We evaluate RULER at the native 32K context length and, after YaRN position extrapolation, at 128K. Benchmark prompts, shot counts, decoding settings, seeds, and metric aggregation are held fixed across attention variants within each model scale.

B DETAILED MODEL-LEVEL RESULTS

Tables 8, 9, and 10 report the per-task results summarized in Table 2.

Table 8: **Knowledge Benchmark Results**. Knowledge evaluation of the scaling-law models at 14B and the continued-trained models at 32B. CoWA produces averages comparable to FullAttn at both scales and remains close across individual tasks.

MODEL	MMLU GEN|5-SHOT	MMLU-PRO 5-SHOT	BBH COT|3-SHOT	HELLASWAG 5-SHOT	OBQA 5-SHOT	WINOGRANDE 5-SHOT	AVG
14B-Non-Thinking, temperature=0.6, top_k=20, top_p=0.95, max_gen_toks=32768, seed=[0, 42, 233, 666, 1234]							
FullAttn	78.21±0.31	**64.96±0.46**	82.59±0.48	**80.64±0.38**	49.4±2.17	78.16±1.14	72.32
MoBA	76.34±0.46	58.73±0.83	82.3±0.41	80.2±0.34	49.7±2.05	76.24±1.45	70.58
DSA	72.72±0.53	59.92±0.62	81.62±0.63	80.01±0.33	49.53±2.24	77.63±1.17	70.23
CoWA (ours)	**79.04±0.28**	64.89±0.43	**83.64±0.51**	80.3±0.31	**49.9±2.01**	**78.48±1.23**	**72.7**
32B-Thinking, temperature=0.6, top_k=20, top_p=0.95, max_gen_toks=32768, seed=[0, 42, 233, 666, 1234]							
FullAttn	77.27±0.34	**68.57±0.41**	88.39±0.54	**83.4±0.37**	55.0±2.23	81.14±1.1	75.62
CoWA (ours)	**77.90±0.31**	68.51±0.47	**89.42±0.61**	83.1±0.28	**55.8±2.22**	**81.72±0.9**	**76.07**

Knowledge performance. Table 8 shows that CoWA preserves performance across the evaluated knowledge benchmarks, with average scores of 72.70 versus 72.32 for FullAttn at 14B and 76.07 versus 75.62 at 32B. The task-level pattern is consistent across the two model scales: CoWA scores higher on MMLU and BBH, while remaining slightly below FullAttn on MMLU-Pro and HellaSwag. These results support retention of the evaluated knowledge capabilities under complementary long-range allocation, with modest differences across individual tasks.

Table 9: **Reasoning Benchmark Results**. Reasoning evaluation of the scaling-law models at 14B and the continued-trained models at 32B. CoWA achieves average scores comparable to FullAttn at both scales.

MODEL	PIQA 5-SHOT	GSM8K COT|4-SHOT	HENDRYCKS-MATH COT|4-SHOT	ARC-C 0-SHOT	AGIEVAL 0-SHOT	GPQA-DIAMOND COT|3-SHOT	AVG
14B-Non-Thinking, temperature=0.6, top_k=20, top_p=0.95, max_gen_toks=32768, seed=[0, 42, 233, 666, 1234]							
FullAttn	**81.63±0.86**	90.32±0.93	**62.46±0.65**	61.95±1.44	**50.56±0.71**	39.87±3.47	64.46
MoBA	77.73±0.93	**91.29±0.97**	39.0±1.38	**62.45±1.7**	48.96±0.87	38.88±3.81	59.71
DSA	80.42±0.77	90.52±0.99	55.99±0.82	62.04±1.29	47.6±0.63	**43.11±4.52**	63.28
CoWA (ours)	81.5±0.72	90.58±0.83	62.31±0.62	62.1±1.62	50.44±0.76	42.33±3.17	**64.87**
32B-Thinking, temperature=0.6, top_k=20, top_p=0.95, max_gen_toks=32768, seed=[0, 42, 233, 666, 1234]							
FullAttn	**85.97±0.88**	93.33±0.69	**94.94±0.82**	69.54±1.34	54.18±0.68	**56.06±3.54**	**75.67**
CoWA (ours)	85.75±0.83	**93.59±0.67**	94.71±0.84	**69.74±1.41**	**54.36±0.65**	55.05±3.56	75.53

Reasoning performance. Table 9 shows that CoWA closely tracks FullAttn across the evaluated reasoning tasks. Average scores are 64.87 versus 64.46 at 14B and 75.53 versus 75.67 at 32B; excluding GPQA-Diamond, all per-task differences are within 0.26 points at both scales. The modest average gain at 14B is largely driven by GPQA-Diamond, which also exhibits relatively large run-to-run variability. At 14B, CoWA additionally retains a Hendrycks-Math score close to FullAttn (62.31 versus 62.46), whereas MoBA and DSA score 39.00 and 55.99, respectively. Overall, these results support comparable performance on the evaluated reasoning benchmarks.

Table 10: **Retrieval Benchmark Results**. RULER evaluation at the native 32K context length and under YaRN extrapolation to 128K. CoWA achieves average scores close to FullAttn at both context lengths. NIAH-S averages NIAH-S-1, NIAH-S-2, and NIAH-S-3; NIAH-MK averages NIAH-MK-1, NIAH-MK-2, and NIAH-MK-3; and RULER-QA averages RULER-QA-SQuAD and RULER-QA-HotpotQA.

MODEL	NIAH-S	NIAH-MK	NIAH-MQ	NIAH-MV	RULER-VT	RULER-CWE	RULER-FWE	RULER-QA	AVG
14B Native 32K Sequence Length, seed=[0, 42, 233, 666, 1234]									
FullAttn	**100±0**	**98.0±0.65**	**98.8±0.68**	95.5±0.72	**99.44±0.2**	**75.7±0.69**	91.67±0.71	56.28±1.2	**89.42**
MoBA	99.8±0.004	94.4±0.63	98.2±0.73	95.1±0.7	99.0±0.32	67.35±0.96	91.44±0.69	52.12±1.64	87.17
DSA	**100±0**	95.6±0.68	98.0±0.77	**96.1±0.76**	99.24±0.26	69.25±0.87	91.74±0.67	51.42±1.43	87.67
CoWA (ours)	**100±0**	96.96±0.71	98.7±0.69	95.2±0.73	99.28±0.21	72.6±0.8	**92.53±0.66**	**57.74±1.14**	89.13
32B Native 32K Sequence Length, seed=[0, 42, 233, 666, 1234]									
FullAttn	**100±0**	**99.87±0.09**	**99.8±0.08**	**99.4±0.13**	**99.64±0.18**	**85.18±0.51**	94.53±0.53	63.2±0.93	**92.7**
CoWA (ours)	**100±0**	98.93±0.11	99.69±0.09	99.17±0.17	99.49±0.2	83.26±0.59	**95.42±0.45**	**64.64±0.88**	92.58
14B YaRN 128K Sequence Length, seed=[0, 42, 233, 666, 1234]									
FullAttn	**98.4±0.51**	55.2±1.64	**88.25±0.69**	64.1±1.16	**93.04±0.63**	5.76±0.46	**84.26±0.79**	**37.73±2.15**	65.84
MoBA	94.8±0.57	53.2±1.77	83.35±0.83	62.4±1.27	90.2±0.71	0.03±0.02	80.53±0.92	27.67±2.33	61.52
DSA	97.5±0.6	55.4±1.62	84.1±0.8	64.0±1.14	92.62±0.61	5.42±0.86	80.07±1.07	33.34±1.87	64.05
CoWA (ours)	97.06±0.56	**57.6±1.44**	84.6±0.79	**66.65±1.11**	92.88±0.63	**13.22±1.12**	83.27±0.8	37.54±2.04	**66.60**
32B YaRN 128K Sequence Length, seed=[0, 42, 233, 666, 1234]									
FullAttn	**99.8±0.12**	**86.4±1.35**	**95.7±0.56**	88.0±0.79	**98.76±0.22**	45.68±1.03	**93.27±1.25**	**48.61±2.17**	**82.03**
CoWA (ours)	99.6±0.15	84.3±1.47	95.1±0.6	**88.5±0.81**	98.53±0.25	**48.23±0.94**	91.84±1.31	48.11±2.13	81.78

Long-context performance. Table 10 shows that CoWA retains aggregate retrieval performance comparable to FullAttn at both the native 32K context length and under YaRN extrapolation to 128K. At 128K, its average score is 66.60 versus 65.84 for FullAttn at 14B, and 81.78 versus 82.03 at 32B. Individual tasks exhibit both gains and losses. Some differences are small relative to the reported run-to-run evaluation variability, whereas others are more pronounced, such as the lower NIAH-MQ score and higher RULER-CWE score at 14B. These results support comparable aggregate long-context performance, while indicating task-specific trade-offs rather than uniform preservation of every retrieval capability.