PHBA: PREFIX-STATE HYBRID BLOCK ATTENTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Hybrid architectures combining linear sequence models with softmax attention provide an effective balance between efficient long-context modeling and precise token retrieval. Existing designs such as Native Hybrid Attention (NHA) combine compressed long-term states with sliding-window attention, but their exact attention is restricted to a fixed local window. In this work, we introduce Prefix-State Hybrid Block Attention (PHBA), which replaces local sliding-window attention with top-k block-sparse retrieval and couples each retrieved block with a compact prefix state summarizing its preceding context. The prefix states are constructed by a gated linear recurrence at block boundaries and retrieved together with the corresponding token blocks, allowing the model to combine precise long-range evidence with compressed historical context within a unified layer. We further develop a hardware-aware Triton implementation that streams routed token blocks and prefix states without materializing large intermediate tensors. Experiments show that PHBA improves long-context and retrieval performance over strong linear and hybrid baselines while retaining efficient training and inference.

1 INTRODUCTION

Long-context language modeling requires both efficient sequence processing and reliable retrieval of information from distant positions. Standard self-attention (Vaswani et al., 2017; Dao et al., 2022) provides precise token-level access to the entire history, but its quadratic complexity makes it increasingly expensive as the context length grows. Linear and recurrent sequence models (Katharopoulos et al., 2020; Choromanski et al., 2020; Gu et al., 2021; Smith et al., 2022; Sun et al., 2023; Gu & Dao, 2023; Dao & Gu, 2024; Yang et al., 2023; Arora et al., 2024; Hu et al., 2025; Qin et al., 2024; Zhang et al., 2024; Team et al., 2025; Hatamizadeh et al., 2026) offer a more efficient alternative by compressing historical information into fixed-size states , but this compression can limit precise in-context retrieval.

A practical solution is to combine recurrent sequence modeling with sparse or local softmax attention. Recent hybrid architectures mix efficient recurrent computation with token-level attention to improve the trade-off between long-range efficiency and recall (Arora et al., 2024; De et al., 2024; Du et al., 2026). For example, Native Hybrid Attention (NHA) (Du et al., 2026) combines a compressed recurrent state with sliding-window attention (SWA), allowing each layer to access both summarized long-term memory and precise recent tokens. However, the exact-attention component in SWA is inherently local: key–value pairs outside the fixed window are accessible only through the compressed recurrent state. As a result, the exact-attention budget cannot be allocated according to the content relevance between each query and the global history. This leaves a central challenge for long-context hybrid models:

> *Can sparse attention retrieve the globally relevant key–value content for each query while preserving its preceding context?*

Sparse attention has long been explored as a way to extend token-level interaction beyond dense quadratic attention (Child et al., 2019; Kitaev et al., 2020; Beltagy et al., 2020; Zaheer et al., 2020; Roy et al., 2021). A natural way to extend exact attention beyond a fixed local window is content-dependent block retrieval. Top-K block-sparse attention allocates the sparse computation to relevant blocks from distant history rather than only to recent tokens. In particular, Mixture of Block Atten-

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.

tion (MoBA) (Lu et al., 2026) uses query-dependent block-level routing to identify a small number of historical blocks for each query.

Based on this observation, we introduce Prefix-State Hybrid Block Attention (PHBA). PHBA replaces the fixed sliding-window component of conventional recurrent–attention hybrids with content-dependent block-sparse retrieval. In addition, PHBA associates each block with a compact prefix state constructed by a gated slot recurrence (Yang et al., 2023; Zhang et al., 2024) at the corresponding block boundary. When a query selects a historical block, it accesses both the exact tokens within that block and its aligned prefix state, which summarizes the causal history preceding the block. PHBA therefore couples high-resolution retrieved content with compressed context immediately preceding that content.

The key design is to align recurrent memory with the same block structure used for sparse routing. Rather than relying on a single compressed history together with fixed local attention, PHBA exposes block-aligned prefix memories that can be selected together with distant exact-token blocks. This allows the model to recover broader historical context around retrieved content without maintaining a separate recurrent state for every token.

Efficiently implementing this computation requires processing dynamically routed token blocks together with their associated prefix states without materializing packed sparse payloads. We therefore develop a hardware-aware Triton implementation that directly streams selected token and state blocks from their original memory layouts and uses compact routing metadata for efficient sparse execution. Combined with chunkwise prefix-state construction and online-softmax accumulation, the implementation supports efficient long-context training and inference. Experiments demonstrate that PHBA consistently improves retrieval and long-context modeling over strong recurrent, sparse-attention, and hybrid baselines while maintaining favorable efficiency at long sequence lengths.

2 PRELIMINARY

PHBA builds upon two components: block-sparse attention for selective token retrieval and gated slot memory for recurrent sequence modeling. This section briefly reviews these two components and introduces the notation used in PHBA.

2.1 BLOCK-SPARSE ATTENTION

Sparse attention reduces the cost of standard self-attention by allowing each query to attend to only a subset of historical tokens. Block-sparse attention groups the sequence into contiguous blocks and performs selection at the block level.

Let the sequence be partitioned into blocks $\{\mathcal{B}_1, \ldots, \mathcal{B}_B\}$, and let $\mathcal{R}_t$ denote the set of blocks selected for query $\boldsymbol{q}_t$. The query attends to the tokens contained in the selected blocks:

$$\boldsymbol{o}_t^{\mathrm{blk}} = \mathrm{softmax}\left(\frac{\boldsymbol{q}_t \boldsymbol{K}_{\mathcal{R}_t}^\top}{\sqrt{d}}\right) \boldsymbol{V}_{\mathcal{R}_t}, \tag{1}$$

where $\boldsymbol{K}_{\mathcal{R}_t}$ and $\boldsymbol{V}_{\mathcal{R}_t}$ contain the key and value vectors from the selected blocks.

Content-dependent block routing, as used in Mixture of Block Attention (MoBA), selects a small number of relevant blocks for each query. PHBA adopts this block-routed sparse-attention setting; the routing procedure is described in Sec. 3.

2.2 GATED SLOT MEMORY

Gated Slot Attention (GSA) (Zhang et al., 2024) represents historical information using a fixed number of key and value memory slots. Let $\widetilde{\boldsymbol{K}}_t \in \mathbb{R}^{M \times d}$ and $\widetilde{\boldsymbol{V}}_t \in \mathbb{R}^{M \times d_v}$ denote the key and value memories, and let $\boldsymbol{A}_t \in (0,1)^M$ denote a slot-wise retention gate. Using the coupled write gate $\boldsymbol{I}_t = \mathbf{1} - \boldsymbol{A}_t$, the memories are updated as

$$\begin{aligned} \widetilde{\boldsymbol{K}}_t &= \mathrm{Diag}(\boldsymbol{A}_t)\widetilde{\boldsymbol{K}}_{t-1} + \boldsymbol{I}_t \boldsymbol{k}_t^\top, \\ \widetilde{\boldsymbol{V}}_t &= \mathrm{Diag}(\boldsymbol{A}_t)\widetilde{\boldsymbol{V}}_{t-1} + \boldsymbol{I}_t \boldsymbol{v}_t^\top. \end{aligned} \tag{2}$$

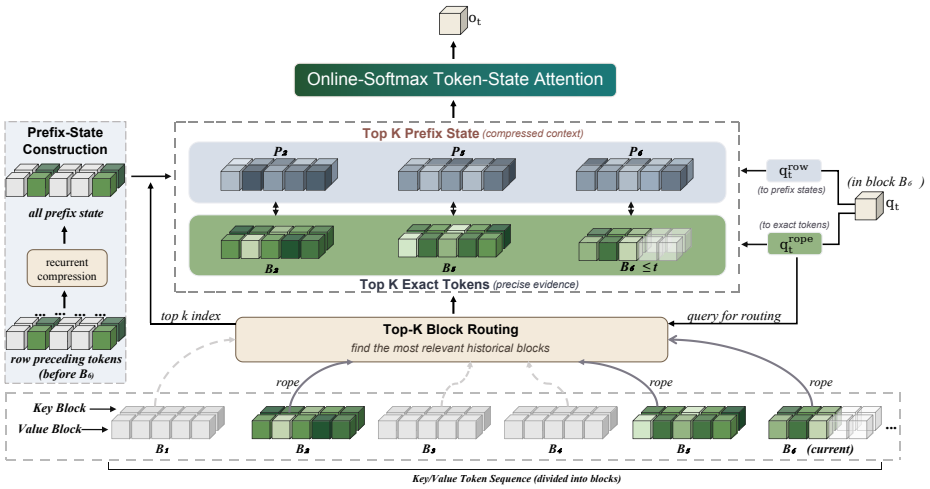


Figure 1: Overview of PHBA. The RoPE query $\boldsymbol{q}_t^{\mathrm{rope}}$ first performs **Top-K Block Routing** to select relevant historical blocks. PHBA attends to the retrieved exact tokens with $\boldsymbol{q}_t^{\mathrm{rope}}$ and their **Block-Aligned Prefix States** with $\boldsymbol{q}_t^{\mathrm{raw}}$, and combines both memory types through **Online-Softmax Token–State Attention** to produce $\boldsymbol{o}_t$.

The shared slot-wise gate keeps the key and value memories temporally aligned. This recurrence is a specialization of gated linear attention (GLA) (Yang et al., 2023) and admits the same hardware-efficient chunkwise-parallel formulation. PHBA adopts this gated slot-memory construction to form block-aligned prefix states, but uses them differently from GSA: the resulting key/value slots are stored only at block boundaries and later participate in joint softmax attention together with routed exact-token blocks.

3 Prefix-State Hybrid Block Attention

Prefix-State Hybrid Block Attention (PHBA) combines block-sparse retrieval with recurrent prefix states within a single attention layer, as illustrated in Figure 1. For each query, PHBA retrieves a small number of relevant historical blocks at full token resolution, while compact recurrent states summarize the prefixes preceding these blocks. The exact token representations and their aligned prefix states share a common softmax normalization, which is evaluated efficiently through online-softmax merging without explicitly materializing a combined attention matrix. We describe PHBA in four parts: top-K block routing, prefix-state construction, online-softmax token–state attention, and efficient execution. Unless otherwise stated, we use single-head notation. Let $\boldsymbol{Q}^{\mathrm{raw}}, \boldsymbol{K}^{\mathrm{raw}} \in \mathbb{R}^{T\times d}$ and $\boldsymbol{V} \in \mathbb{R}^{T\times d_v}$ denote the projected queries, keys, and values. RoPE is applied to the token-attention queries and keys, producing $\boldsymbol{Q}^{\mathrm{rope}}$ and $\boldsymbol{K}^{\mathrm{rope}}$.

3.1 Top-K Block Routing

PHBA partitions the sequence into $N_c = \lceil T/C \rceil$ contiguous blocks of at most C tokens. Let $\mathcal{B}_b$ denote block b, and let $c(t)$ denote the block containing query token t. Historical routing considers only blocks preceding the current block.

PHBA uses a total exact-block budget of K blocks per query. Since the current block is always retained for local causal attention, $K_h = K - 1$ blocks remain for historical routing.

Each block is represented for routing by the mean of its RoPE-transformed keys. For query t, PHBA selects the K_h historical blocks whose summaries have the largest similarity to the query:

$$\bar{\boldsymbol{k}}_b^{\mathrm{rope}} = \frac{1}{|\mathcal{B}_b|}\sum_{i\in\mathcal{B}_b} \boldsymbol{k}_i^{\mathrm{rope}}, \qquad \mathcal{R}_t = \mathrm{TopK}\left(\left\{(\boldsymbol{q}_t^{\mathrm{rope}})^\top \bar{\boldsymbol{k}}_b^{\mathrm{rope}}\right\}_{b<c(t)}, K_h\right). \tag{3}$$

The centroid is used only for routing. Once a historical block is selected, PHBA attends to its original token-level keys and values. The current block is not included in the historical route; instead, local

causal attention over the current block forms part of the token branch. Therefore, the exact-token memory visible to query t is

$$\mathcal{E}_t^{\mathrm{tok}} = \mathcal{B}_{c(t)}^{\leq t} \cup \bigcup_{b \in \mathcal{R}_t} \mathcal{B}_b, \tag{4}$$

where $\mathcal{B}_{c(t)}^{\leq t}$ contains the causal tokens in the current block. This provides local causal context together with query-dependent access to distant historical blocks.

3.2 Prefix-State Construction

Sparse retrieval provides exact token information from selected blocks, while recurrent prefix states provide a compact summary of earlier history. PHBA constructs separate prefix-key and prefix-value states using the gated recurrence introduced in Sec. 2.2.

Let M denote the number of recurrent slots. For token t, PHBA predicts a retention gate $\boldsymbol{A}_t = \sigma(\boldsymbol{a}_t)^{1/\tau} \in (0,1)^M$ and uses the coupled write gate $\boldsymbol{I}_t = \mathbf{1} - \boldsymbol{A}_t$. The prefix states are updated as

$$\begin{aligned} \boldsymbol{S}_t^K &= \mathrm{Diag}(\boldsymbol{A}_t)\boldsymbol{S}_{t-1}^K + \boldsymbol{I}_t(\boldsymbol{k}_t^{\mathrm{raw}})^\top, \\ \boldsymbol{S}_t^V &= \mathrm{Diag}(\boldsymbol{A}_t)\boldsymbol{S}_{t-1}^V + \boldsymbol{I}_t\boldsymbol{v}_t^\top, \end{aligned} \tag{5}$$

where $\boldsymbol{S}_t^K \in \mathbb{R}^{M\times d}$ and $\boldsymbol{S}_t^V \in \mathbb{R}^{M\times d_v}$. The two recurrences share the same slot-wise retention and write gates, ensuring that each prefix-key slot and its corresponding prefix-value slot summarize the same historical positions with identical temporal weighting.

For a block $\mathcal{B}_b$ beginning at position τ_b, PHBA stores the states immediately before the block:

$$\boldsymbol{P}_b^K = \boldsymbol{S}_{\tau_b-1}^K, \qquad \boldsymbol{P}_b^V = \boldsymbol{S}_{\tau_b-1}^V. \tag{6}$$

Thus, $(\boldsymbol{P}_b^K, \boldsymbol{P}_b^V)$ summarizes the causal prefix preceding $\mathcal{B}_b$ and does not contain tokens from the block itself.

For query t, the candidate prefix-state blocks are

$$\mathcal{E}_t^{\mathrm{st}} = \{c(t)\} \cup \mathcal{R}_t. \tag{7}$$

States associated with blocks having an empty causal prefix are masked. The current-block state summarizes all history before the local block, while each routed state provides the prefix associated with the corresponding historical evidence block.

3.3 Online-Softmax Token–State Attention

PHBA attends jointly to two types of memory: exact token representations and block-aligned prefix states. The token branch uses RoPE-transformed queries and keys, whereas the prefix-state branch uses the raw query and the raw prefix-state keys. For a token $i \in \mathcal{E}_t^{\mathrm{tok}}$ and a prefix-state slot m from block $b \in \mathcal{E}_t^{\mathrm{st}}$, the corresponding attention logits are

$$\ell_{t,i}^{\mathrm{tok}} = \frac{(\boldsymbol{q}_t^{\mathrm{rope}})^\top \boldsymbol{k}_i^{\mathrm{rope}}}{\sqrt{d}}, \qquad \ell_{t,b,m}^{\mathrm{st}} = \frac{(\boldsymbol{q}_t^{\mathrm{raw}})^\top \boldsymbol{p}_{b,m}^K}{\sqrt{d}}, \tag{8}$$

where $\boldsymbol{p}_{b,m}^K$ and $\boldsymbol{p}_{b,m}^V$ denote the m-th key and value slots of the prefix state associated with block b.

Although the two memory types are evaluated through separate sparse branches, they are normalized jointly. Define their branch-wise partition functions as

$$Z_t^{\mathrm{tok}} = \sum_{i \in \mathcal{E}_t^{\mathrm{tok}}} e^{\ell_{t,i}^{\mathrm{tok}}}, \qquad Z_t^{\mathrm{st}} = \sum_{b \in \mathcal{E}_t^{\mathrm{st}}} \sum_{m=1}^{M} e^{\ell_{t,b,m}^{\mathrm{st}}}. \tag{9}$$

The PHBA output is

$$\boldsymbol{o}_t^{\mathrm{PHBA}} = \frac{\displaystyle\sum_{i \in \mathcal{E}_t^{\mathrm{tok}}} e^{\ell_{t,i}^{\mathrm{tok}}} \boldsymbol{v}_i + \sum_{b \in \mathcal{E}_t^{\mathrm{st}}} \sum_{m=1}^{M} e^{\ell_{t,b,m}^{\mathrm{st}}} \boldsymbol{p}_{b,m}^V}{Z_t^{\mathrm{tok}} + Z_t^{\mathrm{st}}}. \tag{10}$$

Importantly, PHBA does not explicitly concatenate the token and prefix-state memories. Instead, each branch maintains its own online-softmax statistics. Let $\boldsymbol{O}_t^{\mathrm{tok}}, L_t^{\mathrm{tok}}$ and $\boldsymbol{O}_t^{\mathrm{st}}, L_t^{\mathrm{st}}$ denote the normalized outputs and log-sum-exp statistics of the token and prefix-state branches, respectively. The two branches are merged exactly as

$$\begin{aligned} L_t &= \mathrm{LSE}\left(L_t^{\mathrm{tok}}, L_t^{\mathrm{st}}\right), \\ \boldsymbol{O}_t &= e^{L_t^{\mathrm{tok}} - L_t} \boldsymbol{O}_t^{\mathrm{tok}} + e^{L_t^{\mathrm{st}} - L_t} \boldsymbol{O}_t^{\mathrm{st}}. \end{aligned} \tag{11}$$

This online-softmax merge is exactly equivalent to applying a single softmax over the union of exact tokens and prefix-state slots, while avoiding explicit concatenation of the two memory types. Consequently, each query allocates probability mass directly between exact token evidence and compressed prefix information, rather than combining independently normalized branch outputs through an external fusion weight. In implementation, the two branch outputs need not be materialized separately: the same online-softmax merge is applied incrementally while streaming the selected token and prefix-state blocks. The hardware-efficient realization is described in Sec. 3.4 and Algorithm 6.

3.4 Hardware-Efficient Chunkwise and Sparse Execution

PHBA addresses two sources of training inefficiency. First, the recurrent prefix-state construction is reformulated into a chunkwise-parallel computation, avoiding token-wise recurrent execution within each chunk. Second, routed attention is implemented with metadata-driven sparse kernels that avoid gathering selected queries and memory blocks into packed payload tensors. Together, these two designs enable efficient training while preserving the PHBA formulation in Sec. 3.3.

Chunkwise Prefix-State Update. The prefix-state recurrence follows the gated slot-memory structure of GSA and admits the same chunkwise decomposition as gated linear attention. We use the same block partition as in Sec. 3.1. For packed inputs, the recurrence is applied independently to each sequence and the incoming state is reset to zero at its first block. Let $C_n = |\mathcal{B}_n|$ denote the length of block n. For block $[n]$, define

$$\overrightarrow{\boldsymbol{A}}_{[n],i} = \prod_{j=1}^{i} \boldsymbol{A}_{[n],j}, \qquad \overleftarrow{\boldsymbol{A}}_{[n],i} = \prod_{j=i+1}^{C_n} \boldsymbol{A}_{[n],j}. \tag{12}$$

where the products are elementwise. For compact notation, we write $\boldsymbol{S}_{[n]} = [\boldsymbol{S}_{[n]}^K, \boldsymbol{S}_{[n]}^V]$ and $\boldsymbol{Z}_{[n]} = [\boldsymbol{K}_{[n]}^{\mathrm{raw}}, \boldsymbol{V}_{[n]}]$. Let $\boldsymbol{I}_{[n]} = \mathbf{1} - \boldsymbol{A}_{[n]}$ and $\boldsymbol{W}_{[n]} = \boldsymbol{I}_{[n]} \odot \overleftarrow{\boldsymbol{A}}_{[n]}$. The boundary state is updated as

$$\boldsymbol{S}_{[n+1]} = \mathrm{Diag}\left(\overrightarrow{\boldsymbol{A}}_{[n],C_n}\right) \boldsymbol{S}_{[n]} + \boldsymbol{W}_{[n]}^{\top} \boldsymbol{Z}_{[n]}. \tag{13}$$

Only the boundary state is recurrent across chunks, while the writes within each chunk are evaluated in parallel. The incoming state $\boldsymbol{S}_{[n]}$ is stored as the prefix state associated with chunk $[n]$. Thus, PHBA retains the causal recurrent update while exposing only block-aligned prefix states to the subsequent sparse attention computation. A detailed derivation of Eq. 13 is provided in Appendix A.1.1.

Operator-Aware Sparse Execution. PHBA is designed around a simple systems constraint: routed attention should not first gather selected query rows and memory blocks into packed sparse payload tensors. Such a pipeline would introduce additional HBM round trips before the attention kernel can start. PHBA instead materializes only compact integer routing metadata and directly loads the required query rows, token blocks, and prefix-state blocks from their original HBM layouts into on-chip SRAM/registers.

Algorithm 1 first produces the query-centric historical routes used by the forward pass. During training, the same routing decisions are additionally reformatted into block-major variable-length metadata for the backward pass, without moving the underlying payloads. For the forward pass, block-local causal FlashAttention first computes row-wise outputs and log-sum-exp statistics. The fused sparse kernel then extends the corresponding online-softmax accumulators with the current-block prefix state, routed historical token blocks, and their aligned prefix states. This is exactly equivalent to the joint token–state softmax in Eq. 10 and the online-softmax merge in Eq. 11, without explicitly concatenating the two memory types. Algorithm 2 summarizes the forward execution.

Algorithm 1 PHBA Routing and Block-Major Metadata Construction

Require: RoPE queries/keys $\boldsymbol{Q}^{\mathrm{rope}}, \boldsymbol{K}^{\mathrm{rope}} \in \mathbb{R}^{T\times H\times d}$ in HBM. Boundaries cu. Block size C. Total exact-block budget K.

1: Set historical routing budget $K_h \leftarrow K - 1$.
2: Construct current block ids $\boldsymbol{c} \in \mathbb{Z}^T$, block-row pointers $\mathrm{BlockPtr} \in \mathbb{Z}^{N_c+1}$, and prefix-validity mask $\mathrm{PrefixValid} \in \{0,1\}^{N_c}$ from cu and C, where $\mathrm{PrefixValid}[b] = 1$ iff block b has a nonempty causal prefix.
3: Compute key-block centroids $\bar{\boldsymbol{K}}^{\mathrm{rope}} \in \mathbb{R}^{N_c\times H\times d}$ using Algorithm 3.
4: Select historical top-K_h blocks $\boldsymbol{R}^c \in \mathbb{Z}^{T\times H\times K_h}$ using Algorithm 4.
5: Set historical-token routes $\boldsymbol{R}^{\mathrm{tok}} \leftarrow \boldsymbol{R}^c$.
6: Broadcast $\boldsymbol{c}$ over heads as $\boldsymbol{c}^H \in \mathbb{Z}^{T\times H\times 1}$ and set prefix-state routes $\boldsymbol{R}^{\mathrm{st}} \leftarrow [\boldsymbol{c}^H; \boldsymbol{R}^c]$ along the route dimension.
7: Set any entry in $\boldsymbol{R}^{\mathrm{st}}$ whose routed block b has $\mathrm{PrefixValid}[b] = 0$ to -1.
8: During training, reformat $\boldsymbol{R}^{\mathrm{tok}}$ and $\boldsymbol{R}^{\mathrm{st}}$ into block-major varlen metadata for sparse backward using Algorithm 5.
9: **return** $\boldsymbol{R}^c, \boldsymbol{c}, \mathrm{BlockPtr}, \mathrm{PrefixValid}$ and, during training, $(\boldsymbol{\rho}^{\mathrm{tok}}, \mathrm{Offset}^{\mathrm{tok}}, C^{\mathrm{tok}}_{\mathrm{blk}})$, $(\boldsymbol{\rho}^{\mathrm{st}}, \mathrm{Offset}^{\mathrm{st}}, C^{\mathrm{st}}_{\mathrm{blk}})$.

Algorithm 2 PHBA Forward Pass

Require: Token tensors $\boldsymbol{Q}^{\mathrm{raw}}, \boldsymbol{Q}^{\mathrm{rope}}, \boldsymbol{K}^{\mathrm{rope}} \in \mathbb{R}^{T\times H\times d}$ and $\boldsymbol{V} \in \mathbb{R}^{T\times H\times d_v}$ in HBM. Prefix states $\boldsymbol{P}^K \in \mathbb{R}^{N_c\times H\times M\times d}$ and $\boldsymbol{P}^V \in \mathbb{R}^{N_c\times H\times M\times d_v}$ in HBM. Routes $\boldsymbol{R}^c$, current block ids $\boldsymbol{c}$, block-row pointers BlockPtr, and prefix-validity mask PrefixValid. Tile sizes B_r, B_c.

1: Run block-local causal FlashAttention according to BlockPtr and write $\boldsymbol{O}^{\mathrm{loc}} \in \mathbb{R}^{T\times H\times d_v}$ and $L^{\mathrm{loc}} \in \mathbb{R}^{T\times H}$ to HBM.
2: Run FUSEDSPARSEFORWARD (Algorithm 6) with $(\boldsymbol{Q}^{\mathrm{raw}}, \boldsymbol{Q}^{\mathrm{rope}}, \boldsymbol{K}^{\mathrm{rope}}, \boldsymbol{V}, \boldsymbol{P}^K, \boldsymbol{P}^V)$, routes $(\boldsymbol{R}^c, \boldsymbol{c})$, BlockPtr, PrefixValid, seed $(\boldsymbol{O}^{\mathrm{loc}}, L^{\mathrm{loc}})$, and tile sizes B_r, B_c.
3: The fused kernel extends the corresponding online-softmax accumulators with the current-block prefix state, historical exact-token blocks, and their aligned prefix states.
4: The fused kernel writes final output $\boldsymbol{O}$ and joint log-sum-exp L to HBM.
5: **return** $\boldsymbol{O}, L$.

Detailed centroid computation, top-K selection, metadata reformatting, sparse forward, and backward kernels are provided in Appendix A.1.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Models and Training. We evaluate PHBA at two scales: 760M models trained on 50B tokens and 1.3B models trained on 100B tokens. All models are trained from scratch on FineWeb-Edu (Penedo et al., 2024) with an 8K context length. Unless otherwise specified, PHBA uses $M = 128$ prefix-state slots and block size $C = 128$. For the 1.3B main comparison, we use a fixed $1/8$ exact-token budget, corresponding to $K = 8$ blocks at 8K. Additional architecture, optimization, and training details are provided in Appendix A.2.1.

Baselines. We compare PHBA with Transformer (Vaswani et al., 2017), MoBA (Lu et al., 2026), GSA (Zhang et al., 2024), KDA (Team et al., 2025), and NHA (Du et al., 2026). All baselines are trained on the same data and context length at matched model scale; efficient baselines use matched token budgets where applicable. Detailed baseline information is provided in Appendix A.2.2.

Evaluation. We evaluate language modeling, zero-shot commonsense reasoning, long-context retrieval, and real-world understanding. Language modeling and commonsense evaluation use `lm-evaluation-harness` (Gao et al., 2024). Models trained at 8K are directly evaluated at 16K–64K without further training. We use RULER/NIAH for long-context retrieval and LongBench for real-world understanding, truncating LongBench inputs beyond 8K to the first and last 4K tokens. Detailed benchmark descriptions are provided in Appendix A.2.3.

Table 1: Short-context evaluation of 760M and 1.3B models. RULER/NIAH are evaluated at 4K and 8K and scaled by 100. **Bold**/underline denote best/second-best results. PHBA (default): $M = C = 128$, $K = 8$.

PART I: Common Sense Reasoning & RULER Tasks

Model & Scale	Common Sense Reasoning						RULER Tasks											
	ARC_e (acc↑)	ARC_c (acc↑)	Hella. (acc↑)	PIQA (acc↑)	Wino. (acc↑)	Avg. (acc↑)	CWE 4k↑	CWE 8k↑	FWE 4k↑	FWE 8k↑	HotpotQA 4k↑	HotpotQA 8k↑	SQuAD 4k↑	SQuAD 8k↑	VT 4k↑	VT 8k↑	Avg 4k↑	Avg 8k↑
760M params with 50B training tokens																		
PHBA (origin)	66.96	34.30	52.14	70.73	55.33	55.89	26.3	3.6	**19.5**	**22.1**	20.2	15.6	24.4	14.6	1.0	**2.0**	18.3	11.6
w. $C = 512, K = 2$	64.10	32.34	48.25	69.70	52.25	53.33	4.8	0.3	1.6	3.4	23.0	16.4	26.6	10.0	**10.9**	0.6	13.4	6.1
w. $C = 256, K = 4$	64.35	32.00	49.89	69.70	53.67	53.92	20.0	0.6	4.1	9.2	21.0	15.8	24.3	9.7	9.0	0.4	15.7	7.1
w. $C = 64, K = 16$	**67.51**	34.22	**53.25**	**71.38**	55.88	**56.45**	**34.7**	**5.5**	11.0	3.9	24.4	**22.2**	**37.8**	**22.1**	0.9	0.3	**21.8**	10.8
w. $M = 64$	66.08	32.59	51.70	70.40	55.64	55.28	10.1	3.1	2.9	2.7	**24.6**	19.0	33.1	19.6	1.8	0.0	14.5	8.9
w. $M = 256$	64.94	31.31	48.85	70.13	55.96	54.24	21.4	4.0	0.0	0.0	23.4	18.8	27.9	12.9	4.8	0.1	15.5	7.2
w/o. gate	64.52	**34.39**	49.98	70.24	53.75	54.58	30.5	1.1	4.1	2.1	20.0	17.0	21.9	10.9	0.6	0.0	15.4	6.2
w. mean	65.61	33.53	50.19	70.24	**57.38**	55.39	18.1	0.1	7.5	4.1	20.8	17.8	24.9	15.2	7.4	0.8	15.7	7.6
1.3B params with 100B training tokens																		
Transformer	72.14	38.74	57.41	71.33	58.33	59.59	**38.8**	2.2	25.7	27.2	**30.2**	24.6	34.3	25.6	5.6	0.0	26.9	15.9
MoBA	70.16	36.09	56.25	71.38	55.96	57.97	35.2	1.6	18.2	3.9	25.2	21.0	37.4	22.0	0.8	5.5	23.4	10.8
GSA	68.94	35.84	56.15	72.31	56.67	57.98	1.5	0.4	1.3	2.0	17.2	15.8	14.4	10.1	**32.4**	**23.6**	13.4	10.4
KDA	**73.53**	**40.02**	**58.89**	73.45	59.43	**61.06**	28.6	3.5	41.8	33.3	23.6	21.6	23.4	14.1	7.6	10.8	25.0	16.7
NHA	71.42	36.09	58.35	**74.37**	58.48	59.75	33.5	7.0	**45.7**	23.1	11.8	**26.2**	19.2	**27.2**	21.8	1.3	26.4	15.1
PHBA	71.17	38.14	58.58	72.63	**59.83**	60.07	31.0	**24.2**	31.3	**45.3**	29.0	13.4	**43.4**	16.2	1.5	20.5	**27.2**	**23.9**

PART II: Language Modeling Tasks

760M params with 50B training tokens				1.3B params with 100B training tokens			
Model	Wiki. (ppl↓)	Lamb. (acc↑)	Lamb. (ppl↓)	Model	Wiki. (ppl↓)	Lamb. (acc↑)	Lamb. (ppl↓)
PHBA (origin)	18.33	42.98	16.33	Transformer	15.67	46.69	12.26
w. $C = 512, K = 2$	19.76	40.31	19.34	MoBA	16.22	47.22	13.04
w. $C = 256, K = 4$	19.19	40.52	17.86	GSA	16.86	40.93	16.74
w. $C = 64, K = 16$	**17.34**	**45.24**	**14.60**	KDA	16.18	48.65	10.91
w. $M = 64$	18.84	43.74	15.91	NHA	15.96	48.34	11.51
w. $M = 256$	19.74	41.90	18.15	PHBA	**15.21**	**49.27**	**10.80**
w/o. gate	18.88	43.31	16.11	**Table Color Legend**			
w. mean	18.76	43.24	16.38	**Blue:** Proposed Method **Gray:** Primary Baseline **Red:** Average Columns			

PART III: NIAH Tasks

Model & Scale	Single-Key NIAH						Multi-Key NIAH						NIAH (Extra)				Avg	
	Task 1 4k↑	Task 1 8k↑	Task 2 4k↑	Task 2 8k↑	Task 3 4k↑	Task 3 8k↑	Task 1 4k↑	Task 1 8k↑	Task 2 4k↑	Task 2 8k↑	Task 3 4k↑	Task 3 8k↑	MultiQuery 4k↑	MultiQuery 8k↑	MultiValue 4k↑	MultiValue 8k↑	Mean 4k↑	Mean 8k↑
760M params with 50B training tokens																		
PHBA (origin)	**99.4**	20.4	97.4	42.8	**67.4**	11.4	78.0	44.0	**43.2**	5.6	**18.8**	0.0	41.7	22.2	**46.0**	25.0	**61.5**	21.4
w. $C = 512, K = 2$	78.6	12.6	36.2	19.6	25.4	3.6	32.0	20.4	1.4	0.6	1.0	0.2	23.1	13.9	23.1	15.8	27.6	10.8
w. $C = 256, K = 4$	88.0	6.2	66.6	33.2	35.2	6.0	43.2	26.0	5.6	2.4	12.6	0.6	27.1	19.0	29.3	20.3	38.5	14.2
w. $C = 64, K = 16$	99.2	52.0	**98.8**	**61.6**	57.6	**12.2**	**82.8**	**51.8**	24.2	**5.8**	11.6	**1.6**	**44.8**	20.8	42.9	**25.3**	57.7	**28.9**
w. $M = 64$	99.0	20.2	93.4	50.4	41.4	12.0	72.2	42.2	10.2	1.4	9.4	0.2	32.7	**23.0**	39.0	23.2	49.7	21.6
w. $M = 256$	66.2	3.6	75.4	23.2	13.2	4.6	42.6	18.4	2.6	0.2	7.6	0.0	29.7	9.3	31.4	11.9	33.6	8.9
w/o. gate	92.6	37.4	65.2	21.0	32.8	5.6	36.0	16.2	2.6	0.6	4.8	0.4	30.5	13.5	24.9	14.0	36.2	13.6
w. mean	94.8	**53.8**	68.6	36.6	4.2	6.0	50.0	35.6	9.0	0.4	4.4	0.0	29.4	17.3	29.9	20.8	36.3	21.3
1.3B params with 100B training tokens																		
Transformer	99.8	72.4	**100.0**	**73.6**	**63.2**	**42.6**	**84.2**	**68.0**	**75.8**	**40.2**	12.0	**1.0**	**57.9**	**31.3**	43.2	26.8	**67.0**	**44.5**
MoBA	98.8	64.0	83.0	37.4	15.8	10.6	41.8	32.8	7.2	3.2	8.6	0.0	30.0	19.0	29.5	21.4	39.3	23.6
GSA	93.0	60.8	88.0	20.2	2.4	1.0	24.2	21.6	0.0	0.0	0.0	0.0	34.4	14.2	34.4	14.0	34.6	16.5
KDA	**100.0**	29.6	86.2	49.2	46.0	19.4	26.4	53.0	0.2	14.0	0.0	0.4	42.7	27.1	49.3	**28.0**	43.8	27.5
NHA	**100.0**	98.6	95.0	48.2	53.8	10.8	30.0	24.8	40.0	0.0	0.0	0.0	38.2	19.4	39.1	17.4	49.5	27.4
PHBA	**100.0**	**99.4**	98.6	39.6	54.4	23.6	76.0	26.6	45.8	0.0	**13.0**	0.0	55.8	20.5	**59.0**	26.1	62.6	29.5

4.2 Short-Context Language Modeling and Retrieval

Table 1 reports performance within the 8K training context.

Language Modeling and Commonsense Reasoning. Table 1 reports WikiText and LAMBADA (Merity et al., 2016; Paperno et al., 2016) together with zero-shot ARC, HellaSwag, PIQA, and WinoGrande (Clark et al., 2018; Zellers et al., 2019; Bisk et al., 2020; Sakaguchi et al., 2021). At 1.3B scale, PHBA achieves the best WikiText and LAMBADA results among the compared models and a competitive commonsense average of 60.07, second only to KDA. This shows that routed retrieval and prefix-state memory preserve strong short-context modeling quality.

Short-Context Retrieval. On RULER (Hsieh et al., 2024) and its NIAH variants, PHBA consistently outperforms the other efficient baselines at 4K and 8K, trailing only the dense Transformer on NIAH. This demonstrates strong retrieval within the training context.

4.3 Long-Context Retrieval and Understanding

We next evaluate the 1.3B models beyond the 8K training length. Figure 2 (a)–(c) reports RULER, NIAH, and LongBench averages, with complete results in Appendix Table 2.

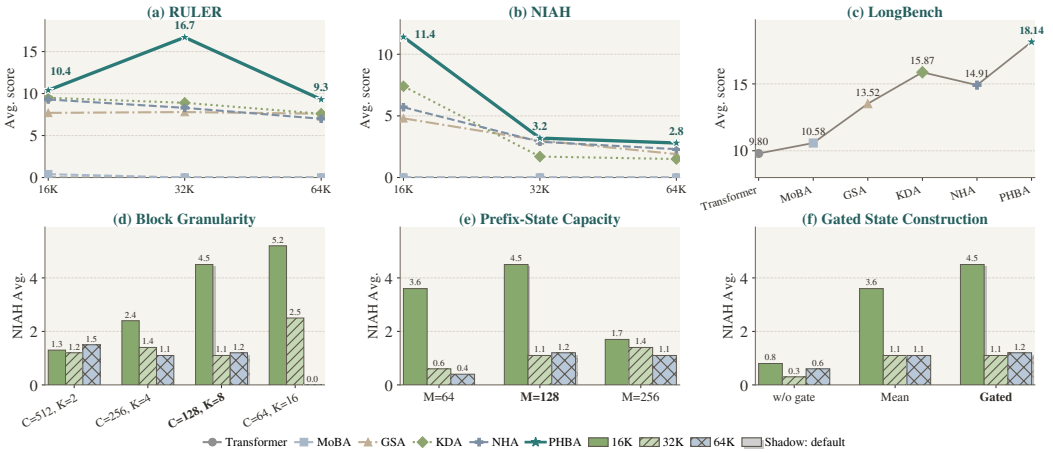


Figure 2: Long-context results and ablations. (a)–(b) RULER and NIAH averages at 16K, 32K, and 64K; (c) LongBench average. (d)–(f) Ablations of block granularity, prefix-state capacity, and gated state construction. Bars show 16K/32K/64K results; shadowed groups mark the default settings.

Long-Context Retrieval. PHBA maintains the strongest overall RULER and NIAH (Hsieh et al., 2024) performance from 16K to 64K, with the advantage becoming more pronounced beyond the training length. Content-dependent routing preserves direct access to distant token-level information, while aligned prefix states provide the context preceding the retrieved blocks, improving long-context extrapolation over compressed memory or sparse token retrieval alone.

Long-Context Understanding. PHBA also achieves the strongest LongBench (Bai et al., 2024) average, with gains across document QA, summarization, and few-shot tasks while remaining competitive on code. This indicates that the retrieval gains transfer beyond controlled synthetic settings to real-world long-context tasks.

4.4 Ablation and Design Analysis

Figure 2 (d)–(f) summarizes the long-context ablations of block granularity, prefix-state capacity, and gated state construction. Short-context ablations are included in Table 1, with complete long-context results in Appendix Table 2.

Block Granularity. Under a fixed $1/8$ exact-token budget, reducing C and increasing K generally improves retrieval by enabling finer-grained routing (Figure 2 (d); Table 1). Smaller blocks, however, require more routed blocks and reduce throughput. We therefore use $C = 128$ and $K = 8$ as the default quality–efficiency trade-off.

Prefix-State Capacity. Figure 2 (e) and Table 1 show that performance does not improve monotonically with state size. $M = 128$ gives the strongest and most stable results across context lengths, while smaller states lack capacity and larger states provide no consistent benefit.

Gated State Construction. Figure 2 (f) and Table 1 show that the learned gated recurrence consistently outperforms ungated and mean-aggregation variants, especially under long-context extrapolation. This highlights the importance of adaptive retention and writing for prefix-state construction.

Token–State Composition. MoBA retrieves distant tokens without an aligned summary of their preceding context, whereas NHA combines recurrent memory with exact attention restricted to a fixed local window. PHBA instead couples routed historical blocks with their block-aligned prefix states, yielding stronger retrieval and long-context performance in Figure 2 (a)–(c) and Table 1.

4.5 Training Efficiency.

Figure 3 compares end-to-end training throughput, with detailed benchmark settings and measurement protocols provided in Sec. A.2.4. With a fixed 64-token window, NHA is about 24% slower

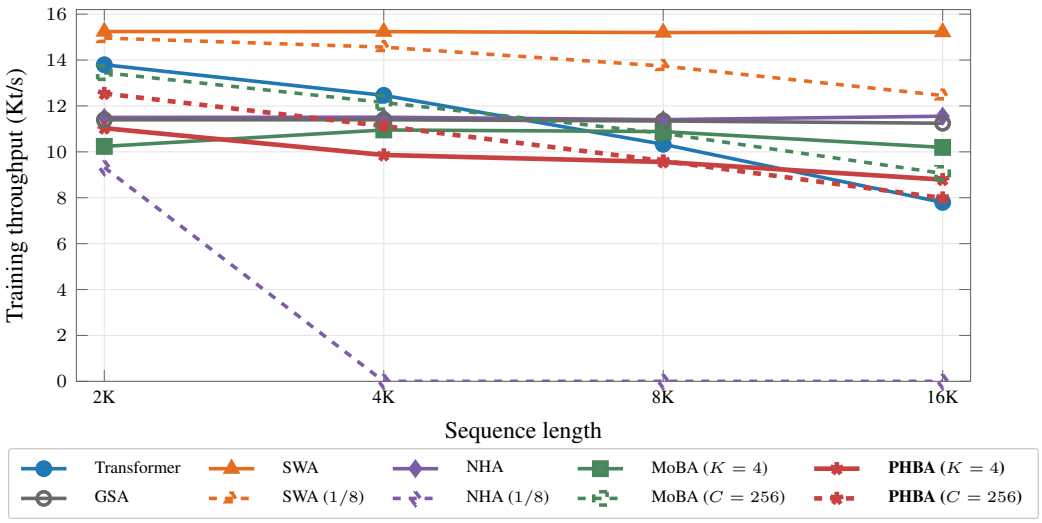


Figure 3: Training efficiency on a single NVIDIA H20-96GB GPU. For MoBA and PHBA, both configurations maintain a fixed $1/8$ sparse token budget: $K = 4$ dynamically adjusts the block size C, whereas $C = 256$ dynamically adjusts the number of routed blocks K. SWA $(1/8)$ and NHA $(1/8)$ similarly scale the window size with sequence length, while SWA and NHA use a fixed window size of 64. All state-based models use $M = 128$ memory slots. Zero throughput for NHA $(1/8)$ denotes out-of-memory (OOM).

than SWA. Under the matched $1/8$ budget, PHBA is 18.4–34.6% faster than NHA at 2K, while NHA runs out of memory from 4K onward in the evaluated implementation. Relative to MoBA, adding block-aligned prefix states incurs only 6.7–11.9% overhead for $C = 256$ across 2K–16K. With $K = 4$, PHBA reaches 8.79 Kt/s at 16K, 12.8% higher than full attention, while GSA remains nearly length-independent due to its recurrent computation.

4.6 Related Work

Efficient sequence models replace dense attention with structured or recurrent state updates, including S4/S5, RetNet, Mamba/Mamba-2, GLA, HGRN2, GSA, and recent gated or delta-rule variants such as DeltaNet, KDA, and Gated DeltaNet-2 (Gu et al., 2021; Smith et al., 2022; Sun et al., 2023; Gu & Dao, 2023; Dao & Gu, 2024; Yang et al., 2023; Qin et al., 2024; Zhang et al., 2024; Yang et al., 2024; Team et al., 2025; Hatamizadeh et al., 2026). A complementary line reduces softmax-attention cost through sparse connectivity, from Sparse Transformer, Reformer, Longformer, BigBird, and Routing Transformer to MoBA's query-dependent block routing (Child et al., 2019; Kitaev et al., 2020; Beltagy et al., 2020; Zaheer et al., 2020; Roy et al., 2021; Lu et al., 2026). Hybrid architectures such as BASED, Griffin, and NHA combine recurrent or linear memory with local attention (Arora et al., 2024; De et al., 2024; Du et al., 2026); NHA further places recurrent key–value slots and sliding-window tokens under a shared softmax. In this work, PHBA builds on these lines by pairing content-routed historical blocks with block-aligned prefix states, so each selected block is accompanied by a compressed summary of its preceding context.

5 Conclusion

We introduced Prefix-State Hybrid Block Attention (PHBA), which combines content-dependent block-sparse retrieval with block-aligned recurrent prefix states for efficient long-context modeling. PHBA jointly attends to routed historical tokens and the compressed context preceding them, providing both fine-grained retrieval and broader historical information within a unified softmax. A hardware-aware implementation further enables efficient chunkwise state construction and sparse execution without materializing packed payloads. Experiments demonstrate consistent improvements in retrieval and long-context understanding while preserving competitive language modeling quality and training efficiency.

AI USE STATEMENT

In this work, we used generative AI tools solely to aid and polish the writing. We did not use generative AI tools for research design, methodology, experiments, data analysis, or the generation of results. All AI-assisted content was reviewed by the authors. We take full responsibility for the final content of this work.

ETHICS STATEMENT

This work studies efficient sequence modeling architectures and does not involve human subjects or the collection of sensitive personal data. Experiments are conducted using publicly available training and evaluation datasets. We are not aware of additional ethical concerns specific to the proposed method beyond those generally associated with large language models.

REPRODUCIBILITY STATEMENT

We provide detailed specifications of PHBA in the main paper, including its routing, prefix-state construction, token–state attention, and efficient execution. The appendix contains the chunkwise derivation, complete operator pseudocode, training and evaluation settings, throughput benchmark protocol, and full experimental results. Together, these materials are intended to support faithful reproduction of the proposed method and experiments. The source code and implementation will be made publicly available upon acceptance.

AUTHOR CONTRIBUTIONS

If you'd like to, you may include a section for author contributions as is done in many journals. This is optional and at the discretion of the authors.

ACKNOWLEDGMENTS

Use unnumbered third level headings for the acknowledgments. All acknowledgments, including those to funding agencies, go at the end of the paper.

REFERENCES

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, Dylan Zinsley, James Zou, Atri Rudra, and Christopher Ré. Simple linear attention language models balance the recall-throughput tradeoff. *arXiv preprint arXiv:2402.18668*, 2024.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 3119–3137, 2024.

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

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, pp. 7432–7439, 2020.

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

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

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.

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

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022.

Soham De, Samuel L Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, et al. Griffin: Mixing gated linear recurrences with local attention for efficient language models. *arXiv preprint arXiv:2402.19427*, 2024.

Jusen Du, Jiaxi Hu, Zhang Tao, Weigao Sun, and Yu Cheng. Native hybrid attention for efficient sequence modeling. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3826–3842, 2026.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL `https://zenodo.org/records/12608602`.

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

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

Ali Hatamizadeh, Yejin Choi, and Jan Kautz. Gated deltanet-2: Decoupling erase and write in linear attention. *arXiv preprint arXiv:2605.22791*, 2026.

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.

Jiaxi Hu, Yongqi Pan, Jusen Du, Disen Lan, Xiaqiang Tang, Qingsong Wen, Yuxuan Liang, and Weigao Sun. Comba: Improving bilinear rnns with closed-loop control. *arXiv preprint arXiv:2506.02475*, 2025.

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

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, et al. Moba: Mixture of block attention for long-context llms. *Advances in Neural Information Processing Systems*, 38:17790–17815, 2026.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc-Quan 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 (volume 1: Long papers)*, pp. 1525–1534, 2016.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.

Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. Hgrn2: Gated linear rnns with state expansion. *arXiv preprint arXiv:2404.07904*, 2024.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021.

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.

Jimmy TH Smith, Andrew Warrington, and Scott W Linderman. Simplified state space layers for sequence modeling. *arXiv preprint arXiv:2208.04933*, 2022.

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

Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, et al. Kimi linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2023.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. *Advances in neural information processing systems*, 37:115491–115522, 2024.

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*, pp. 4791–4800, 2019.

Yu Zhang, Songlin Yang, Ruijie Zhu, Yue Zhang, Leyang Cui, Yiqiao Wang, Bolun Wang, Freda Shi, Bailin Wang, Wei Bi, et al. Gated slot attention for efficient linear-time sequence modeling. *Advances in Neural Information Processing Systems*, 37:116870–116898, 2024.

A Appendix

A.1 PHBA Operator Pseudocode

This appendix expands the PHBA kernels in Section 3.4. We write C for the token block size, M for the number of prefix-state slots, and H for the number of heads. A logical block is either a token block or a prefix-state block. A physical tile is the smaller query, memory, or feature tile loaded from HBM to SRAM/registers by one Triton program.

The important implementation choice is that PHBA does not build packed sparse payload tensors. The forward kernel consumes the query-centric historical routes $\boldsymbol{R}^c$ directly and loads routed token and prefix-state blocks from their original HBM layouts. During training, the same routing decisions are reformatted into block-major integer metadata $\boldsymbol{\rho}, \mathrm{Offset}, C_{\mathrm{blk}}$ for efficient sparse backward. The generic symbols $\boldsymbol{\rho}, \mathrm{Offset}, C_{\mathrm{blk}}$ denote either the historical-token metadata $(\boldsymbol{\rho}^{\mathrm{tok}}, \mathrm{Offset}^{\mathrm{tok}}, C_{\mathrm{blk}}^{\mathrm{tok}})$ or the prefix-state metadata $(\boldsymbol{\rho}^{\mathrm{st}}, \mathrm{Offset}^{\mathrm{st}}, C_{\mathrm{blk}}^{\mathrm{st}})$ from the main text.

A.1.1 CHUNKWISE PARALLEL PREFIX-STATE MEMORY CONSTRUCTION

This subsection derives the chunkwise parallel form of the prefix-state recurrence in Eq. 5. For each token t, PHBA maintains the combined slot-state matrix $\boldsymbol{S}_t = \left[\widetilde{\boldsymbol{K}}_t, \widetilde{\boldsymbol{V}}_t\right] \in \mathbb{R}^{M\times(d+d_v)}$ and writes $\boldsymbol{z}_t = \left[\boldsymbol{k}_t^{\mathrm{raw}}, \boldsymbol{v}_t\right] \in \mathbb{R}^{d+d_v}$. The recurrent update is

$$\boldsymbol{S}_t = \mathrm{Diag}(\boldsymbol{A}_t)\boldsymbol{S}_{t-1} + \boldsymbol{I}_t\boldsymbol{z}_t^\top, \tag{14}$$

where $\boldsymbol{A}_t \in (0,1)^M$ is the retention gate and $\boldsymbol{I}_t = \boldsymbol{1} - \boldsymbol{A}_t$ is the coupled write gate. PHBA separates memory formation from sparse readout: the recurrence constructs boundary prefix key/value memories, and the attention layer later selects among them.

We now derive the chunkwise parallel form. For packed inputs, the recurrence is reset at each sequence boundary specified by cu, so the first block of every sequence receives a zero incoming state. For any token-wise tensor $\boldsymbol{X}$, let $\boldsymbol{X}_{[n]}$ denote its rows in chunk n, where $C_n = |\mathcal{B}_n|$. Thus $\boldsymbol{K}_{[n]}^{\mathrm{raw}} \in \mathbb{R}^{C_n\times d}$, $\boldsymbol{V}_{[n]} \in \mathbb{R}^{C_n\times d_v}$, and $\boldsymbol{A}_{[n]}, \boldsymbol{I}_{[n]} \in \mathbb{R}^{C_n\times M}$ stack the raw keys, values, retention gates, and write gates in this chunk. Define $\boldsymbol{Z}_{[n]} = \left[\boldsymbol{K}_{[n]}^{\mathrm{raw}}, \boldsymbol{V}_{[n]}\right] \in \mathbb{R}^{C_n\times(d+d_v)}$. Partially unrolling Eq. 14 for the r-th row inside chunk n gives

$$\boldsymbol{S}_{[n],r} = \mathrm{Diag}\left(\prod_{i=1}^{r}\boldsymbol{A}_{[n],i}\right)\boldsymbol{S}_{[n],0} + \sum_{i=1}^{r}\left(\boldsymbol{I}_{[n],i}\odot\prod_{j=i+1}^{r}\boldsymbol{A}_{[n],j}\right)\boldsymbol{z}_{[n],i}^\top, \tag{15}$$

where $\boldsymbol{z}_{[n],i} = \left[\boldsymbol{k}_{[n],i}^{\mathrm{raw}}, \boldsymbol{v}_{[n],i}\right]$. Here $\boldsymbol{S}_{[n],0}$ is the boundary state entering chunk n, and $\boldsymbol{S}_{[n],C_n}$ is the boundary state leaving it. For brevity, we write $\boldsymbol{S}_{[n]} \equiv \boldsymbol{S}_{[n],0}$ and $\boldsymbol{S}_{[n+1]} \equiv \boldsymbol{S}_{[n],C_n}$.

As in chunkwise GLA, define the cumulative retention products

$$\overrightarrow{\boldsymbol{A}}_{[n],r} = \prod_{j=1}^{r}\boldsymbol{A}_{[n],j}, \qquad \overleftarrow{\boldsymbol{A}}_{[n],r} = \prod_{j=r+1}^{C_n}\boldsymbol{A}_{[n],j}. \tag{16}$$

All products over the gate vectors are elementwise. The forward product $\overrightarrow{\boldsymbol{A}}$ propagates boundary memory across the chunk, while the backward product $\overleftarrow{\boldsymbol{A}}$ decays each within-chunk write to the chunk boundary. Let $\overleftarrow{\boldsymbol{A}}_{[n]} \in \mathbb{R}^{C_n\times M}$ stack these backward products. The compact write matrix is

$$\boldsymbol{W}_{[n]} = \boldsymbol{I}_{[n]}\odot\overleftarrow{\boldsymbol{A}}_{[n]} \in \mathbb{R}^{C_n\times M}, \qquad \boldsymbol{\Delta}_{[n]} = \boldsymbol{W}_{[n]}^\top\boldsymbol{Z}_{[n]} \in \mathbb{R}^{M\times(d+d_v)}. \tag{17}$$

Then the chunkwise boundary update becomes

$$\boldsymbol{S}_{[n+1]} = \underbrace{\mathrm{Diag}\left(\overrightarrow{\boldsymbol{A}}_{[n],C_n}\right)\boldsymbol{S}_{[n]}}_{\text{inter-chunk recurrence}} + \underbrace{\boldsymbol{\Delta}_{[n]}}_{\text{intra-chunk parallel write}}. \tag{18}$$

Equation 18 is the form used for parallel training. Only the boundary state $\boldsymbol{S}_{[n]}$ is sequential across chunks. Inside each chunk, the writes reduce to matrix products over chunk rows and state slots. Thus PHBA materializes one prefix key/value state per chunk boundary rather than one recurrent state per token. For numerical stability, the implementation forms these cumulative retention factors from chunk-local prefix sums of $\log \boldsymbol{A}$ rather than explicitly multiplying gate values.

A.1.2 ROUTING METADATA KERNELS

PHBA uses a three-stage routing pipeline: compute block centroids, select top-K_h historical block ids with a tiled score kernel, and reformat the query-centric routes into block-major metadata for sparse backward. The dense query–centroid score matrix is never materialized to HBM. The main routing procedure also constructs $\mathrm{PrefixValid} \in \{0,1\}^{N_c}$, where $\mathrm{PrefixValid}[b] = 1$ iff block b has a nonempty causal prefix.

Block Centroids. Algorithm 3 computes one routing centroid for each block and attention head. Blocks are formed independently within each packed sequence specified by cu, and only valid rows contribute to the centroid.

Algorithm 3 Fused Key-Block Centroid Computation

Require: RoPE keys $\boldsymbol{K}^{\mathrm{rope}} \in \mathbb{R}^{T\times H\times d}$ in HBM. Block-row pointers $\mathrm{BlockPtr} \in \mathbb{Z}^{N_c+1}$ induced by cu and block size C.

Allocate centroid tensor $\bar{\boldsymbol{K}}^{\mathrm{rope}} \in \mathbb{R}^{N_c\times H\times d}$ in HBM.
for each block j and head h **in parallel do**
  Set $\mathcal{B}_j = [\mathrm{BlockPtr}[j], \mathrm{BlockPtr}[j+1])$.
  Stream valid rows of $\boldsymbol{K}^{\mathrm{rope}}_{\mathcal{B}_j,h}$ from HBM through SRAM and accumulate their sum in fp32 registers.
  Divide by $|\mathcal{B}_j|$ on chip and write $\bar{\boldsymbol{K}}^{\mathrm{rope}}_{j,h}$ to HBM.
end for
return $\bar{\boldsymbol{K}}^{\mathrm{rope}}$.

Historical Block Selection. Algorithm 4 scans centroid tiles and maintains a running top-K_h list for each query and head. Only blocks from the same packed sequence and strictly preceding the query's current block are eligible. If fewer than K_h blocks are available, unused route entries remain -1. The discrete route indices are treated as non-differentiable metadata.

Algorithm 4 Tiled Top-K Historical Block Selection

Require: RoPE queries $\boldsymbol{Q}^{\mathrm{rope}} \in \mathbb{R}^{T\times H\times d}$ and centroids $\bar{\boldsymbol{K}}^{\mathrm{rope}} \in \mathbb{R}^{N_c\times H\times d}$ in HBM. Boundaries cu. Block-row pointers $\mathrm{BlockPtr}$. Current block ids $\boldsymbol{c}$. Total exact-block budget K. Tile sizes B_r, B_c.

Set historical routing budget $K_h \leftarrow K - 1$.
For each head and packed sequence, partition query rows and centroid rows into tiles of sizes B_r and B_c, respectively.
Initialize $\boldsymbol{R}^c \in \mathbb{Z}^{T\times H\times K_h}$ to -1 in HBM.
for each query tile i and head h **in parallel do**
  Load $\boldsymbol{Q}^{\mathrm{rope}}_{i,h}$ from HBM to SRAM and initialize top-K_h scores to $-\infty$ and block ids to -1.
  for each centroid tile j in the same packed sequence **do**
    Load $\bar{\boldsymbol{K}}^{\mathrm{rope}}_{j,h}$ from HBM to SRAM.
    Compute $\boldsymbol{S}_{ij,h} \leftarrow \boldsymbol{Q}^{\mathrm{rope}}_{i,h}(\bar{\boldsymbol{K}}^{\mathrm{rope}}_{j,h})^\top$ on chip.
    Apply row-wise $b < c(t)$ and valid-row masks and update the running top-K_h distinct block ids in registers.
  end for
  Write the final block ids to $\boldsymbol{R}^c$ in HBM.
end for
return $\boldsymbol{R}^c$.

Block-Major Metadata. Algorithm 5 inverts the query-centric routing relation. It counts query–head rows assigned to each logical block–head pair, prefix-sums the counts to obtain contiguous ranges, and scatters flattened query–head row ids into these ranges. Only integer metadata is materialized; token and state payloads remain in their original HBM layouts.

A.1.3 Query-Tile Fused Sparse Forward Kernel

Algorithm 6 implements the PHBA forward pass directly from the query-centric routes $\boldsymbol{R}^c$. Each query tile starts from the block-local causal FlashAttention output and log-sum-exp, then extends the same online-softmax accumulators with the current-block prefix state, routed historical token blocks, and their aligned prefix states.

Within a query tile, the B_rK_h historical route entries use a fixed-capacity schedule rather than dynamic compaction. A leader mask marks the first valid occurrence of each historical block id, allowing one loaded block to serve all rows in the tile that route to it. The token branch uses $(\boldsymbol{Q}^{\mathrm{rope}}, \boldsymbol{K}^{\mathrm{rope}}, \boldsymbol{V})$, whereas the state branch uses $(\boldsymbol{Q}^{\mathrm{raw}}, \boldsymbol{P}^K, \boldsymbol{P}^V)$. Payload tensors remain in their original HBM layouts.

Algorithm 5 Reformat Routes to Block-Major Varlen Metadata

Require: Query-centric routes $\boldsymbol{R} \in \mathbb{Z}^{T\times H\times K_{\text{sel}}}$ in HBM, with distinct valid block ids within each query–head row. Number of logical blocks N_c. Number of heads H.

Initialize $C_{\text{blk}} \in \mathbb{Z}^{N_c H}$ to zero in HBM.
for each route slot (t, h, k) **in parallel do**
  Set $b \leftarrow \boldsymbol{R}[t, h, k]$.
  if $b \geq 0$ **then**
    $\text{atomicAdd}(\&C_{\text{blk}}[bH + h], 1)$.
  end if
end for
Exclusive prefix-sum C_{blk} to obtain $\text{Offset} \in \mathbb{Z}^{N_c H+1}$ with $\text{Offset}[0] = 0$.
Set $N_{\text{edge}} \leftarrow \text{Offset}[N_c H]$, allocate $\boldsymbol{\rho} \in \mathbb{Z}^{N_{\text{edge}}}$, and initialize $\text{Cursor} \in \mathbb{Z}^{N_c H}$ to zero in HBM.
for each route slot (t, h, k) **in parallel do**
  Set $b \leftarrow \boldsymbol{R}[t, h, k]$.
  if $b \geq 0$ **then**
    Set $e \leftarrow bH + h$, $r \leftarrow tH + h$, and $p \leftarrow \text{atomicAdd}(\&\text{Cursor}[e], 1)$.
    Store r into $\boldsymbol{\rho}[\text{Offset}[e] + p]$.
  end if
end for
return $\boldsymbol{\rho}, \text{Offset}, C_{\text{blk}}$.

A.1.4 Sparse Backward Kernel

The backward pass complements the query-centric forward with a block-major schedule. Algorithm 7 uses $\boldsymbol{\rho}$ and Offset to enumerate the query–head rows accessing each token or prefix-state block. Sparse scores and probabilities are recomputed on chip from the final joint log-sum-exp L, avoiding materialized sparse probability tensors.

For compactness, let $(\boldsymbol{Q}^{\text{tok}}, \boldsymbol{K}^{\text{tok}}_{j,h}, \boldsymbol{V}^{\text{tok}}_{j,h})$ denote $(\boldsymbol{Q}^{\text{rope}}, \boldsymbol{K}^{\text{rope}}_{\mathcal{B}_j,h}, \boldsymbol{V}_{\mathcal{B}_j,h})$, and let $(\boldsymbol{Q}^{\text{st}}, \boldsymbol{K}^{\text{st}}_{j,h}, \boldsymbol{V}^{\text{st}}_{j,h})$ denote $(\boldsymbol{Q}^{\text{raw}}, \boldsymbol{P}^{K}_{j,h}, \boldsymbol{P}^{V}_{j,h})$, where $\mathcal{B}_j = [\text{BlockPtr}[j], \text{BlockPtr}[j+1])$. The two logical memories therefore contain $|\mathcal{B}_j|$ token rows and M prefix-state slots, respectively.

A.2 Additional Experimental Details

A.2.1 Model Architecture and Training.

We train all models from scratch on FineWeb-Edu with a context length of 8K. The 760M and 1.3B models are trained on 50B and 100B tokens, respectively, with hidden dimensions of 1536 and 2048 and a head dimension of 128. The 1.3B models use 16 attention heads and an MLP expansion ratio of 4. Unless otherwise specified, PHBA uses $M = 128$ prefix-state slots and block size $C = 128$. For the 1.3B main comparison, the exact-token budget is fixed to $1/8$ of the sequence. At the 8K training length, this corresponds to 1024 exact tokens per query, or $K = 8$ blocks consisting of the current causal block and seven routed historical blocks.

We use a 32K SentencePiece tokenizer and optimize all models with AdamW, using $\beta_1 = 0.9$, $\beta_2 = 0.95$, a peak learning rate of 3×10^{-4}, weight decay of 0.1, and gradient clipping at 1.0. The learning rate follows cosine decay with a 1% warmup. Training uses a global batch size of 64 sequences (524,288 tokens per optimization step) in bfloat16 on 8 NVIDIA B200-192GB GPUs.

A.2.2 Baseline Configuration.

We compare PHBA with Transformer (Vaswani et al., 2017), MoBA (Lu et al., 2026), GSA (Zhang et al., 2024), KDA (Team et al., 2025), and NHA (Du et al., 2026). To isolate the effect of the sequence-mixing mechanism, all compared models use standalone architectures rather than hybrid layer compositions. All baselines are trained on the same data and context length at matched model scale, while following the recommended architectural configurations of the corresponding methods. Sparse-attention methods use matched exact-token budgets where applicable.

Algorithm 6 Query-Tile Fused Token–State Sparse Forward

Require: $\boldsymbol{Q}^{\mathrm{raw}}, \boldsymbol{Q}^{\mathrm{rope}} \in \mathbb{R}^{T\times H\times d}$, $\boldsymbol{K}^{\mathrm{rope}} \in \mathbb{R}^{T\times H\times d}$, and $\boldsymbol{V} \in \mathbb{R}^{T\times H\times d_v}$ in HBM. Prefix states $\boldsymbol{P}^K, \boldsymbol{P}^V$ in HBM. Routes $\boldsymbol{R}^c$ and current block ids $\boldsymbol{c}$. Block-row pointers $\mathrm{BlockPtr}$. Prefix-validity mask $\mathrm{PrefixValid}$. Seed $\boldsymbol{O}^{\mathrm{loc}}, L^{\mathrm{loc}}$. Tile sizes B_r, B_c.

Divide query rows within each current block and head into contiguous tiles of size B_r.
for each query tile $\mathcal{Q}_i$ **in parallel do**
  Let c_i and h_i be the current block and head shared by rows in $\mathcal{Q}_i$.
  Load $\boldsymbol{Q}_i^{\mathrm{rope}}, \boldsymbol{Q}_i^{\mathrm{raw}}$ and $\boldsymbol{O}_i^{\mathrm{loc}}, L_i^{\mathrm{loc}}$ from HBM.
  Initialize row-wise fp32 online-softmax accumulators as $m_i \leftarrow L_i^{\mathrm{loc}}$, $\ell_i \leftarrow \mathbf{1}$, and $\boldsymbol{U}_i \leftarrow \boldsymbol{O}_i^{\mathrm{loc}}$.
    $\triangleright$ $e^{L_i^{\mathrm{loc}}}\boldsymbol{O}_i^{\mathrm{loc}}$ equals the unnormalized local-attention numerator.
  Load historical routes $\boldsymbol{R}_i^c$ to SRAM, setting padded query rows to -1.
  Set $\mu_i^{\mathrm{cur}} \leftarrow \mathrm{PrefixValid}[c_i]$ and divide the M current-prefix slots into tiles of size B_c.
  for each current-prefix-state tile **do**
    Load the corresponding $\boldsymbol{P}^K_{c_i,h_i}, \boldsymbol{P}^V_{c_i,h_i}$ rows from HBM to SRAM under validity and tail masks.
    Compute scores using $\boldsymbol{Q}_i^{\mathrm{raw}}/\sqrt{d}$, mask invalid scores to $-\infty$, and update $(m_i, \ell_i, \boldsymbol{U}_i)$ on chip.
  end for
  Flatten $\boldsymbol{R}_i^c$ into $\boldsymbol{b}_i = (b_{i,1}, \ldots, b_{i,J})$ with $J = B_r K_h$.
  Compute leader mask $\lambda_{i,p} \leftarrow \mathbb{1}[b_{i,p} \geq 0]\, \mathbb{1}[b_{i,p} \notin \{b_{i,1}, \ldots, b_{i,p-1}\}]$ on chip.
  for each fixed candidate slot $p = 1, \ldots, J$ **do**
    Set $b \leftarrow b_{i,p}$, $\tilde{b} \leftarrow \max(b, 0)$, and $\boldsymbol{\chi}_{i,p}[r] \leftarrow \lambda_{i,p}\, \mathbb{1}[\exists k : \boldsymbol{R}_i^c[r,k] = b]$.
    Set $\mathcal{B}_{\tilde{b}} \leftarrow [\mathrm{BlockPtr}[\tilde{b}], \mathrm{BlockPtr}[\tilde{b}+1])$ and divide it into token-memory tiles of size B_c.
    for each token-memory tile **do**
      Load the corresponding $\boldsymbol{K}^{\mathrm{rope}}_{\mathcal{B}_{\tilde{b}},h_i}, \boldsymbol{V}_{\mathcal{B}_{\tilde{b}},h_i}$ rows from HBM to SRAM under leader and tail masks.
      Compute scores using $\boldsymbol{Q}_i^{\mathrm{rope}}/\sqrt{d}$, mask rows with $\boldsymbol{\chi}_{i,p}[r] = 0$ and invalid tails, and update $(m_i, \ell_i, \boldsymbol{U}_i)$ on chip.
    end for
    Set $\mu^{\mathrm{st}}_{i,p} \leftarrow \lambda_{i,p}\mathrm{PrefixValid}[\tilde{b}]$ and divide the M aligned-prefix slots into tiles of size B_c.
    for each prefix-state tile **do**
      Load the corresponding $\boldsymbol{P}^K_{\tilde{b},h_i}, \boldsymbol{P}^V_{\tilde{b},h_i}$ rows from HBM to SRAM under validity and tail masks.
      Compute scores using $\boldsymbol{Q}_i^{\mathrm{raw}}/\sqrt{d}$, mask rows with $\boldsymbol{\chi}_{i,p}[r] = 0$ or $\mu^{\mathrm{st}}_{i,p} = 0$, and update $(m_i, \ell_i, \boldsymbol{U}_i)$ on chip.
    end for
  end for
  Finalize $\boldsymbol{O}_i \leftarrow \boldsymbol{U}_i/\ell_i$ and $L_i \leftarrow m_i + \log \ell_i$, and write $\boldsymbol{O}_i, L_i$ to HBM.
end for
return output $\boldsymbol{O}$ and log-sum-exp L.

A.2.3 Evaluation Protocol.

For short-context evaluation, we report language modeling on WikiText and LAMBADA (Merity et al., 2016; Paperno et al., 2016), together with zero-shot commonsense reasoning on ARC-Easy/Challenge, HellaSwag, PIQA, and WinoGrande (Clark et al., 2018; Zellers et al., 2019; Bisk et al., 2020; Sakaguchi et al., 2021). All short-context evaluations use `lm-evaluation-harness` (Gao et al., 2024).

For long-context evaluation, models trained only at 8K are directly evaluated at 16K, 32K, and 64K without further training or fine-tuning. We use RULER (Hsieh et al., 2024) to evaluate retrieval and reasoning over long sequences, including common-word extraction (CWE), question answering based on HotpotQA and SQuAD, and controlled NIAH variants. We further evaluate real-world long-context understanding on LongBench (Bai et al., 2024), covering document QA, summariza-

Algorithm 7 Metadata-Driven Sparse Backward

Require: Token tensors $\boldsymbol{Q}^{\mathrm{raw}}, \boldsymbol{Q}^{\mathrm{rope}}, \boldsymbol{K}^{\mathrm{raw}}, \boldsymbol{K}^{\mathrm{rope}}, \boldsymbol{V}$ in HBM. Prefix states $\boldsymbol{P}^K, \boldsymbol{P}^V$ and retention gates $\boldsymbol{A}$. Block-row pointers $\mathrm{BlockPtr}$ and boundaries cu. Final output and gradient $\boldsymbol{O}, \boldsymbol{dO} \in \mathbb{R}^{T\times H\times d_v}$, joint log-sum-exp $L \in \mathbb{R}^{T\times H}$, block-major metadata $(\boldsymbol{\rho}^\beta, \mathrm{Offset}^\beta, C^\beta_{\mathrm{blk}})$ for $\beta \in \{\mathrm{tok}, \mathrm{st}\}$, and tile sizes B_r, B_c.

Compute $D \leftarrow \mathrm{rowsum}(\boldsymbol{dO} \circ \boldsymbol{O})$ and initialize query and memory-gradient buffers to zero.
Run block-local FlashAttention-style backward using the final joint L, D and accumulate into the token-gradient buffers.
for $\beta \in \{\mathrm{tok}, \mathrm{st}\}$ **do**
  for each logical block j and head h with $C^\beta_{\mathrm{blk}}[jH + h] > 0$ **in parallel do**
    Read routed query rows $\mathcal{R}^\beta_{j,h}$ from $\boldsymbol{\rho}^\beta, \mathrm{Offset}^\beta$ and partition $(\boldsymbol{K}^\beta_{j,h}, \boldsymbol{V}^\beta_{j,h})$ into memory tiles of size B_c.
    for each memory tile $\mathcal{M}$ **do**
      Load $\boldsymbol{K}^\beta_{\mathcal{M},h}, \boldsymbol{V}^\beta_{\mathcal{M},h}$ and initialize $\boldsymbol{dK}^\beta_{\mathcal{M},h}, \boldsymbol{dV}^\beta_{\mathcal{M},h} \leftarrow 0$.
      for each B_r-sized query tile in $\mathcal{R}^\beta_{j,h}$ **do**
        Gather $\boldsymbol{Q}^\beta, \boldsymbol{dO}, L, D$ to SRAM.
        Recompute $\boldsymbol{S} \leftarrow \boldsymbol{Q}^\beta(\boldsymbol{K}^\beta_{\mathcal{M},h})^\top/\sqrt{d}$ and mask invalid tail columns.
        Compute $\boldsymbol{\Pi} \leftarrow \exp(\boldsymbol{S} - L)$, $\boldsymbol{d\Pi} \leftarrow \boldsymbol{dO}(\boldsymbol{V}^\beta_{\mathcal{M},h})^\top$, and $\boldsymbol{dS} \leftarrow \boldsymbol{\Pi} \circ (\boldsymbol{d\Pi} - D)$.
        Accumulate $\boldsymbol{dV}^\beta_{\mathcal{M},h} \mathrel{+}= \boldsymbol{\Pi}^\top \boldsymbol{dO}$ and $\boldsymbol{dK}^\beta_{\mathcal{M},h} \mathrel{+}= \boldsymbol{dS}^\top \boldsymbol{Q}^\beta/\sqrt{d}$.
        Atomically scatter-add $\boldsymbol{dSK}^\beta_{\mathcal{M},h}/\sqrt{d}$ to the corresponding $\boldsymbol{dQ}^\beta$ rows.
      end for
      Accumulate $\boldsymbol{dK}^\beta_{\mathcal{M},h}, \boldsymbol{dV}^\beta_{\mathcal{M},h}$ into HBM.
    end for
  end for
end for
Stack direct prefix-state adjoints as $\boldsymbol{dP}_{[n]} \leftarrow [\boldsymbol{dP}^K_{[n]}, \boldsymbol{dP}^V_{[n]}]$.
Traverse blocks in reverse within each packed sequence, with zero propagated adjoint at the terminal boundary.
for each block $[n]$ in reverse order **do**
  Set $\bar{\boldsymbol{S}}_{[n+1]} \leftarrow \boldsymbol{dP}_{[n+1]} + \bar{\boldsymbol{S}}^{\mathrm{prop}}_{[n+1]}$, with $\boldsymbol{dP}_{[n+1]} = 0$ at an unexposed terminal boundary.
  Load the incoming boundary state $\boldsymbol{S}_{[n]} \leftarrow [\boldsymbol{P}^K_{[n]}, \boldsymbol{P}^V_{[n]}]$.
  Recompute the chunk-local cumulative retention factors and $\boldsymbol{W}_{[n]} \leftarrow \boldsymbol{I}_{[n]} \odot \overleftarrow{\boldsymbol{A}}_{[n]}$ from the retained gate values.
  Backpropagate $\bar{\boldsymbol{S}}_{[n+1]}$ once through $\boldsymbol{S}_{[n+1]} \leftarrow \mathrm{Diag}(\overrightarrow{\boldsymbol{A}}_{[n],C_n})\boldsymbol{S}_{[n]} + \boldsymbol{W}^\top_{[n]}\boldsymbol{Z}_{[n]}$, obtaining $\bar{\boldsymbol{S}}^{\mathrm{prop}}_{[n]} \leftarrow \mathrm{Diag}(\overrightarrow{\boldsymbol{A}}_{[n],C_n})\bar{\boldsymbol{S}}_{[n+1]}$ and gradients to $\boldsymbol{Z}_{[n]} = [\boldsymbol{K}^{\mathrm{raw}}_{[n]}, \boldsymbol{V}_{[n]}]$ and $\boldsymbol{A}_{[n]}$, including the coupled write dependence $\boldsymbol{I}_{[n]} = \boldsymbol{1} - \boldsymbol{A}_{[n]}$.
end for
Backpropagate the retention-gate gradients through $\boldsymbol{A} = \sigma(\boldsymbol{a})^{1/\tau}$.
Backpropagate token-path query/key gradients through RoPE, combine them with the raw query/key gradients from the state path, and combine token-attention and state-construction gradients for $\boldsymbol{V}$.
return accumulated token, state, and gate gradients.

tion, few-shot learning, and code-related tasks. LongBench inputs of at most 8K tokens are evaluated in full; longer inputs are truncated by retaining the first 4K and last 4K tokens.

A.2.4 Efficiency Benchmark Protocol.

Due to hardware availability, the efficiency benchmarks are conducted on a single NVIDIA H20-96GB GPU rather than the NVIDIA B200-192GB GPUs used for model training. All compared methods are evaluated on the same H20-96GB hardware and under the same BF16 training configuration. We benchmark end-to-end training throughput at sequence lengths of 2K, 4K, 8K, and 16K, and report throughput in thousands of processed tokens per second (Kt/s). For sparse-attention meth-

ods, we compare configurations under a matched $1/8$ exact-token budget. For MoBA and PHBA, we consider two complementary parameterizations. In the fixed-K setting, $K = 4$ and the block size C is adjusted with sequence length to preserve the $1/8$ token budget. In the fixed-block setting, $C = 256$ and the number of selected blocks K is adjusted accordingly. This keeps the amount of exact token-level attention comparable across sequence lengths while exposing the different routing and block-size trade-offs. For SWA and NHA, we report both a fixed-window configuration with window size 64 and a matched-budget configuration in which the window size grows with sequence length to maintain the same $1/8$ token budget. All state-based methods use $M = 128$ memory slots. Transformer uses full causal attention. Each measurement covers the complete training computation, including forward and backward propagation and parameter updates. Input preparation and data-loading overhead are excluded from the timing so that the comparison reflects model and operator efficiency. Measurements are taken after warm-up iterations to remove initialization effects. NHA with the $1/8$ configuration exceeds the available GPU memory from 4K onward under the evaluated implementation; these cases are reported as OOM (zero throughput in Fig. 3).

Table 2: Long-context evaluation on RULER, NIAH, and LongBench. Metrics are multiplied by 100. **Bold** and underline indicate the best and second-best results within each model scale, respectively. PHBA (origin) uses $M = 128$, $C = 128$, and $K = 8$.

PART I: RULER Tasks (Long Context)

Model & Scale	CWE			HotpotQA			SQuAD			Avg		
	16k↑	32k↑	64k↑	16k↑	32k↑	64k↑	16k↑	32k↑	64k↑	16k↑	32k↑	64k↑
760M params with 50B training tokens												
PHBA (origin)	0.1	0.0	0.0	**12.8**	10.0	6.4	8.8	6.9	**5.5**	7.2	5.6	4.0
w. $C = 512, K = 2$	0.1	0.0	0.0	11.8	10.4	**10.4**	8.0	6.8	**5.5**	6.6	**5.7**	**5.3**
w. $C = 256, K = 4$	0.1	0.0	**0.1**	9.4	10.6	9.8	4.3	4.7	4.3	4.6	5.1	4.7
w. $C = 64, K = 16$	0.1	0.0	0.0	**12.8**	8.8	0.4	**9.1**	**7.1**	0.2	**7.3**	5.3	0.2
w. $M = 64$	**0.2**	0.0	0.0	10.8	**11.4**	9.0	8.9	5.2	5.1	6.6	5.5	4.7
w. $M = 256$	0.1	0.0	0.0	12.2	9.6	9.6	**9.1**	6.7	4.9	7.1	5.4	4.8
w/o. gate	0.1	0.0	0.0	8.2	4.4	3.2	4.6	3.4	1.9	4.3	2.6	1.7
w. mean	**0.2**	0.0	0.0	10.4	11.0	9.2	7.0	5.9	4.2	5.9	5.6	4.5
1.3B params with 100B training tokens												
Transformer	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
MoBA	0.0	0.0	0.0	0.0	0.0	0.0	1.1	0.1	0.0	0.4	0.0	0.0
GSA	0.2	0.0	0.0	14.0	14.8	**13.6**	9.0	8.6	9.2	7.7	7.8	7.6
KDA	0.2	0.0	0.2	15.4	15.0	13.0	**12.9**	**11.7**	9.5	9.5	8.9	7.6
NHA	0.5	0.1	0.0	**16.0**	**15.6**	13.4	11.3	9.2	7.7	9.3	8.3	7.0
PHBA	**5.0**	**26.0**	**4.0**	14.0	13.0	13.4	12.2	11.0	**10.4**	**10.4**	**16.7**	**9.3**

PART II: NIAH Tasks (Long Context)

Model & Scale	Multi-Key NIAH									NIAH (Extra)						Avg		
	Task 1			Task 2			Task 3			MultiQuery			MultiValue			Mean		
	16k	32k	64k	16k	32k	64k	16k	32k	64k	16k	32k	64k	16k	32k	64k	16k	32k	64k
760M params with 50B training tokens																		
PHBA (origin)	10.0	2.2	**3.2**	0.0	0.0	0.0	0.2	0.0	0.0	4.8	1.2	0.3	7.4	1.9	**2.4**	4.5	1.1	1.2
w. $C = 512, K = 2$	3.2	2.2	**3.2**	0.2	0.0	0.0	**0.4**	0.0	0.0	1.3	1.9	**2.6**	1.6	1.7	1.8	1.3	1.2	**1.5**
w. $C = 256, K = 4$	3.6	2.2	**3.2**	0.0	0.0	0.0	0.0	0.0	0.0	2.9	2.3	0.0	5.4	2.6	**2.4**	2.4	1.4	1.1
w. $C = 64, K = 16$	**10.6**	**5.8**	0.0	0.2	0.0	0.0	0.0	0.0	0.0	**6.1**	**2.4**	0.0	**9.2**	**4.3**	0.0	**5.2**	**2.5**	0.0
w. $M = 64$	7.0	2.2	1.8	**0.8**	0.0	0.0	0.0	0.0	0.0	3.9	0.4	0.0	6.1	0.2	0.4	3.6	0.6	0.4
w. $M = 256$	3.2	2.2	**3.2**	0.0	0.0	0.0	**0.4**	0.0	0.0	2.4	2.3	0.0	2.3	2.6	2.1	1.7	1.4	1.1
w/o. gate	2.4	1.4	2.8	0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.0	0.0	1.5	0.1	0.4	0.8	0.3	0.6
w. mean	8.6	3.4	**3.2**	0.2	0.0	0.0	0.0	0.0	0.0	3.3	0.7	0.2	5.9	1.3	2.2	3.6	1.1	1.1
1.3B params with 100B training tokens																		
Transformer	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
MoBA	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
GSA	9.4	5.8	3.8	0.0	0.0	0.0	0.0	0.0	0.0	6.3	4.1	**3.1**	8.3	5.2	2.5	4.8	3.0	1.9
KDA	13.0	2.6	3.2	3.4	**0.2**	0.0	**0.4**	0.0	0.0	9.4	2.4	1.7	11.0	3.4	2.4	7.4	1.7	1.5
NHA	10.0	4.6	5.8	0.0	0.0	0.0	0.0	0.0	0.0	8.9	**4.6**	2.7	9.4	5.4	3.0	5.7	2.9	2.3
PHBA	**13.6**	**8.2**	**6.2**	**20.0**	0.0	0.0	0.0	0.0	0.0	**11.5**	2.1	3.0	**11.9**	**5.8**	**4.7**	**11.4**	**3.2**	**2.8**

PART III: LongBench Tasks

Model & Scale	Single-Doc		Multi-Doc		Summarization			Few-shot		Code		Avg
	MF.	Qasp.	Hotp.	2Wiki.	Gov.	Multi.	SAM.	TREC	Triv.	LCC	Repo.	
760M params with 50B training tokens												
PHBA (origin)	14.82	**6.54**	6.03	8.84	8.77	8.74	19.06	38.67	24.19	12.95	12.90	14.68
w. $C = 512, K = 2$	14.02	4.87	6.08	8.61	6.82	6.43	18.75	32.67	24.11	15.73	17.35	14.13
w. $C = 256, K = 4$	13.96	5.37	6.17	7.86	6.26	8.41	22.10	29.33	21.08	**16.29**	18.11	14.09
w. $C = 64, K = 16$	15.35	6.31	**7.17**	8.61	8.22	6.97	**24.62**	**43.67**	**24.33**	15.13	14.91	**15.94**
w. $M = 64$	**16.50**	5.93	6.05	**9.33**	**10.85**	**10.55**	16.80	37.83	17.83	15.78	18.54	15.09
w. $M = 256$	13.99	5.32	5.90	8.78	8.17	8.75	17.27	40.67	21.71	14.84	15.17	14.60
w/o. gate	12.04	5.62	5.91	**9.33**	7.75	9.64	12.12	24.00	23.71	16.05	14.30	12.77
w. mean	13.71	6.18	5.70	7.70	7.49	7.22	19.56	30.33	22.85	15.85	**19.68**	14.21
1.3B params with 100B training tokens												
Transformer	10.97	5.83	4.21	5.68	7.60	6.74	10.38	23.00	16.15	8.38	8.91	9.80
MoBA	10.01	5.80	3.74	6.34	8.63	7.43	11.47	25.00	16.39	12.07	9.53	10.58
GSA	13.48	5.33	6.43	8.57	6.31	7.37	24.58	19.00	21.78	16.49	19.34	13.52
KDA	13.64	6.26	**7.57**	8.33	7.58	8.77	**27.20**	33.00	22.17	18.64	21.43	15.87
NHA	13.63	6.08	5.41	7.95	5.96	**10.06**	24.77	32.50	13.34	**21.54**	**22.78**	14.91
PHBA	**15.79**	**7.16**	7.49	**9.85**	**12.10**	9.26	25.06	**46.83**	**26.75**	18.39	20.81	**18.14**