Native Hybrid Attention for Efficient Sequence Modeling

Anonymous ACL submission

Abstract

Transformers excel at sequence modeling but face quadratic complexity, while linear attention offers improved efficiency but often compromises recall accuracy over long contexts. In this work, we introduce **Native Hybrid Attention (NHA)**, a novel hybrid architecture of linear and full attention that integrates both intra & inter-layer hybridization into a unified layer design. NHA maintains long-term context in key–value slots updated by a linear RNN, and augments them with short-term tokens from a sliding window. A single `softmax attention` operation is then applied over all keys and values, enabling per-token and per-head context-dependent weighting without requiring additional fusion parameters. The inter-layer behavior is controlled through a single hyperparameter, the sliding window size, which allows smooth adjustment between purely linear and full attention while keeping all layers structurally uniform. Experimental results show that NHA surpasses Transformers and other hybrid baselines on recall-intensive and commonsense reasoning tasks. Furthermore, pretrained LLMs can be structurally **hybridized** with NHA, achieving competitive accuracy while delivering significant efficiency gains. Code is available at https://anonymous.4open.science/r/NHA-CF8B.

1 Introduction

Self-Attention (Vaswani, 2017) has become the primary architecture for sequence modeling due to its exceptional ability to capture long-term dependencies. However, this power comes at a steep computational cost. The self-attention mechanism has a computational complexity of $O(n^2)$ with respect to the sequence length n. This quadratic scaling presents a major obstacle for processing long sequences, a common requirement in domains like long-document analysis (Beltagy et al., 2020; Zaheer et al., 2020) and bioinformatics (Dalla-Torre et al., 2025; Lin et al., 2023).

To address this limitation, research community has largely pursued two divergent paths. The first path involves **sparse attention**. As shown in Fig. 1, such methods compute softmax attention over sparsely selected tokens, thereby reducing the computational cost (Beltagy et al., 2020; Jiang et al., 2023; Lu et al., 2025; Yuan et al., 2025). Among these approaches, Sliding Window Attention (SWA) currently remains the most efficient implementation, which limits attention computations to a local neighborhood. The second direction explores **linear sequence modeling**, such as linear attention models (Sun et al., 2023; Yang et al., 2024; Du et al., 2025; von Oswald et al., 2025) and state space models (Gu and Dao, 2024; Dao and Gu, 2024; Hu et al., 2025). These models achieve remarkable $O(n)$ efficiency by compressing the entire sequence history into a fixed-size state, thereby enabling a global receptive field. While SWA cannot capture tokens beyond its local window, the extreme compression of linear models often results in the loss of precise token information. Given that these two approaches have complementary strengths and weaknesses, intra-layer hybrid models (Munkhdalai et al., 2024; McDermott et al., 2025; Lan et al., 2025) that combine them are a natural next step.

At the same time, a different trade-off motivates another form of hybridization. On one hand, a pure linear model struggles with the theoretical impossibility of perfectly preserving infinite information within a fixed-size memory state. On the other hand, maintaining a full KV cache for every token at every layer of a standard Transformer is not only computationally prohibitive but often unnecessary, as not all layers require the same level of granular global information. This “all-or-nothing" dilemma in information storage across the network depth has motivated the development of inter-layer hybrid models (Ren et al., 2025; Lieber et al., 2024; Glorioso et al., 2024b,a).

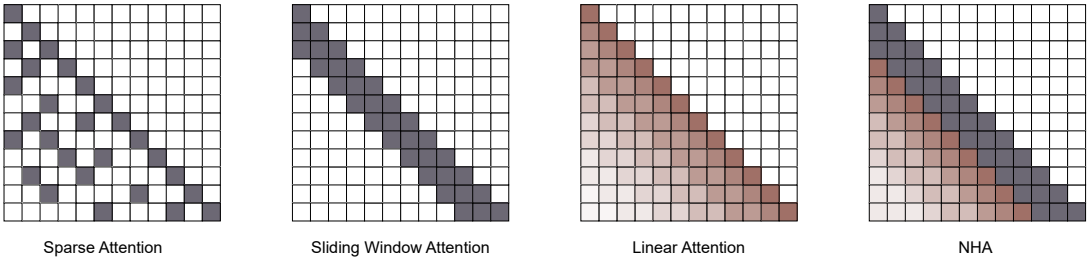

Figure 1: **Token Dependencies.** Illustration for token dependencies of sparse attention, linear attention, and NHA.

These two kinds of hybrid architectures can be summarized as follows:

1. Intra-layer Hybrid: Typically involving computing the linear attention and local softmax attention separately, then combining them through weighted summation.

2. Inter-layer Hybrid: Typically involving stacking or alternating different kinds of layers. For instance, stacking 1 softmax attention layer after every 7 linear attention layers to form an inter-layer hybrid architecture.

In this paper, we introduce **Native Hybrid Attention (NHA)**, *a new hybrid form that natively unifies intra- and inter-layer hybrid architectures into a single, cohesive design*. For intra-layer hybrid, NHA compresses long-term information into a fixed number of slots using a linear RNN model and concatenates them with the local tokens that are inside the window. This combined set of precise, short-term tokens and summarized, long-term slots is then processed by a single, unified `softmax attention` operation. This enables representing long-term memory in the same $(m \times d)$ key–value slot format as SWA, rather than the outer-product form used in many prior hybrids, ensuring compatibility and efficient concatenation.

Unlike prior hybrids that compute separate attentions and fuse them through weighted summation, in NHA, the `softmax attention` mechanism itself learns to dynamically allocate attention between short-term and long-term memory for each query, achieving an automatic and context-aware weighting without the need for manually-tuned or additional parameters. For inter-layer hybrid, all NHA layers share the same design. We can simply change the window size of sliding window attention without modifying the model architecture to control the behavior of each layer. Setting the window size to zero yields a pure linear RNN layer, whereas setting it to the full sequence length recovers a full attention layer. This contrasts with previous inter-layer models that stack different types of layers, which requires managing heterogeneous blocks and their alignment. NHA, therefore, enables flexible, layer-specific configurations without altering the model's fundamental structure.

Our contributions can be summarized as follows:

- **Unified intra- and inter-layer hybrid architecture.** We introduce NHA, which integrates sliding window attention and linear attention within a single, unified `softmax attention` operation, and achieves inter-layer hybridization simply by adjusting the window size, without changing the model architecture. Furthermore, we develop a chunkwise-parallel Triton kernel for efficient GPU computation.

- **Comprehensive evaluation of hybrid and non-hybrid architectures.** We pretrain and evaluate models from each architectural category. Results demonstrate that hybrid architectures consistently surpass standard Transformers on recall-intensive tasks, with NHA attaining the strongest overall performance.

- **Hybridization of pretrained Transformer models.** We demonstrate that NHA can be applied to pretrained Transformer LLMs. With only brief finetuning, these pretrained models can be adapted into the NHA architecture, achieving competitive performance with improved inference speed.

2 Preliminary

To improve the efficiency of Transformers, numerous methods have been developed that restrict the attention window to a fixed number of tokens.

Sliding Window Attention (SWA) SWA (Beltagy et al., 2020) focuses on a fixed-size window of tokens near the current position, storing these tokens precisely. This approach ensures that immediate context is captured accurately but can miss

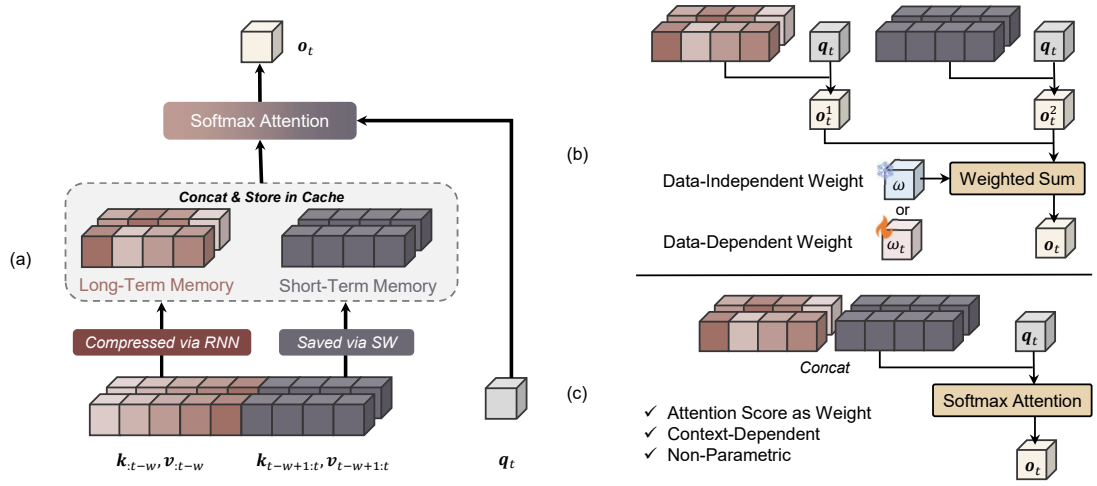

Figure 2: **Intra-Layer Hybrid in NHA**. (a) NHA compresses historical tokens into fixed-size long-term memory slots (brown) through an RNN update, then concatenates them with recent local context (gray) before applying unified `softmax attention`. (b) Previous intra-layer hybrid approaches generally compute long-term and short-term outputs separately and combine them through weighted summation. (c) In contrast, NHA employs a non-parametric, context-dependent `softmax attention` operation to dynamically determine their respective contributions.

important information outside the window, limiting its long-term sequence comprehension.

Gated Slot Attention (GSA) GSA (Zhang et al., 2024b) compresses the entire sequence into a set of fixed memory slots. Instead of storing precise tokens, GSA uses a gating mechanism to update these slots, blending new and past information as in Eq. (1). This compression allows GSA to maintain a broader context and adaptively manage long-term dependencies. Unlike SWA's localized precision focus, GSA balances memory efficiency with comprehensive sequence understanding.

3 Naive Hybrid Attention

3.1 Level 1: Architectural Hybrid of RNN Memory and Softmax Attention

Transformer stores the KV cache for the sequence and performs softmax attention over them to compute the output. In contrast, linear attention compresses the entire sequence into a fixed-size matrix memory and replaces the softmax operation with more efficient matrix multiplication. Prior work (Peng et al., 2021; Zhang et al., 2024b) has explored hybrid designs that compress past tokens into a small set of memory slots and then apply softmax attention over these slots.

$$\begin{aligned} \boldsymbol{K}_t^{\text{long}} &= \text{Diag}(\boldsymbol{\alpha}_t)\, \boldsymbol{K}_{t-1}^{\text{long}} + (1-\boldsymbol{\alpha}_t) \otimes \boldsymbol{k}_t, \\ \boldsymbol{V}_t^{\text{long}} &= \text{Diag}(\boldsymbol{\alpha}_t)\, \boldsymbol{V}_{t-1}^{\text{long}} + (1-\boldsymbol{\alpha}_t) \otimes \boldsymbol{v}_t, \end{aligned} \tag{1}$$

where t represents the current token index, $\boldsymbol{K}_t^{\text{long}}, \boldsymbol{V}_t^{\text{long}} \in \mathbb{R}^{m \times d}$, while m denotes the number of memory slots and $\boldsymbol{\alpha}_t$ denotes an input-dependent tensor of gating value where each element is in the range $[0, 1]$.

However, performing softmax attention solely on this compressed memory does not appear to be an ideal solution, as the precise tokens near the current position often play a crucial role. This limitation motivates us to take it a step further by exploring an intra-layer memory hybrid approach.

3.2 Level 2: Intra-Layer Hybrid

SWA applies softmax attention within a fixed-size window, focusing on the context of nearby tokens. This approach can be seen as primarily leveraging precise local tokens. The KV cache for tokens within the window remains fixed in size and can be regarded as precise short-term memory.

$$\begin{aligned} \boldsymbol{K}_t^{\text{short}} &= \{\boldsymbol{k}_{t-w+1}^T, \boldsymbol{k}_{t-w+2}^T, ..., \boldsymbol{k}_t^T\}^T, \\ \boldsymbol{V}_t^{\text{short}} &= \{\boldsymbol{v}_{t-w+1}^T, \boldsymbol{v}_{t-w+2}^T, ..., \boldsymbol{v}_t^T\}^T, \end{aligned} \tag{2}$$

where t represents the current token index, w denotes the window size, $\boldsymbol{K}_t^{\text{short}}, \boldsymbol{V}_t^{\text{short}} \in \mathbb{R}^{w \times d}$.

Memory Hybridization The long-term memory in Eq. 1 naturally aligns with the short-term memory described above, and the two complement each other effectively. We concatenate them to form a unified, hybrid memory that integrates both long-term and short-term information. Subsequently, the softmax function is used to compute the attention scores, enabling dynamic weighting of long-term and short-term memory.

$$\boldsymbol{K}_t^{\mathrm{H}} = \mathrm{Concat}(\boldsymbol{K}_t^{\mathrm{long}}, \boldsymbol{K}_t^{\mathrm{short}}) \in \mathbb{R}^{(m+w)\times d},$$
$$\boldsymbol{V}_t^{\mathrm{H}} = \mathrm{Concat}(\boldsymbol{V}_t^{\mathrm{long}}, \boldsymbol{V}_t^{\mathrm{short}}) \in \mathbb{R}^{(m+w)\times d}, \tag{3}$$

$$\boldsymbol{o}_t = \mathrm{softmax}(\frac{\boldsymbol{q}_t(\boldsymbol{K}_t^H)^T}{\sqrt{d}})\boldsymbol{V}_t^H, \tag{4}$$

where t represents the current token index, m denotes the number of memory slots, w denotes the window size, and d is the dimension of $\boldsymbol{q}$.

Token shift ensures that only tokens outside the sliding window update the long-term memory. For $t \leq w$, no tokens are removed, and a zero prefix is used to initialize the memory slots. For positional encoding, RoPE is applied within the sliding window, while no positional embedding is added to the long-term memory. Further details and ablation analysis are provided in Appendix C.4.2.

Context-Dependent Fusion In NHA, the unified softmax over concatenated long-term and short-term memory assigns the proportion of attention to the long-term memory as

$$\omega_L = \frac{\sum_{i\in\mathrm{long}} \exp(\boldsymbol{q}_t\boldsymbol{k}_i^\intercal)}{\sum_{i\in\mathrm{long}} \exp(\boldsymbol{q}_t\boldsymbol{k}_i^\intercal) + \sum_{j\in\mathrm{short}} \exp(\boldsymbol{q}_t\boldsymbol{k}_j^\intercal)}, \tag{5}$$

where $\boldsymbol{q}_t$ is the current query, and $\boldsymbol{k}_i, \boldsymbol{k}_j$ are keys from long- and short-term memory respectively. This proportion is computed jointly with token-level weights and depends on similarity scores with all keys in both memory types, thus incorporating information from all preceding tokens.

To further highlight the distinction between unified softmax and output-level weighted fusion, we provide a gradient analysis in Appendix D and a memory analysis in Appendix E. This shows that unified softmax naturally couples gradient flow between long- and short-term memories. We compare the performance of unified softmax fusion and learnable weighted fusion in ablations (Sec. 4.7).

Since the window size can vary across different layers in the model, it creates the possibility for an inter-layer hybrid. This flexibility leads us to a deeper exploration of model architecture.

3.3 Level 3: Inter-Layer Hybrid via Window Size Adjustment

For tasks requiring precise retrieval of long past sequences, pure linear models struggle as it is infeasible to retain all information in a fixed-size memory as the sequence length increases. Storing every $\boldsymbol{kv}$ in each layer, as transformers do, appears excessive. This trade-off motivates an inter-layer hybrid approach: strategically placing full attention mechanisms only in select layers, using linear models to **sparsify $\boldsymbol{kv}$ storage across the depth of the Transformer**. This leverages the efficiency of linear models while retaining the robust memory capabilities of attention where most needed. NHA is inherently suited for this hybrid strategy.

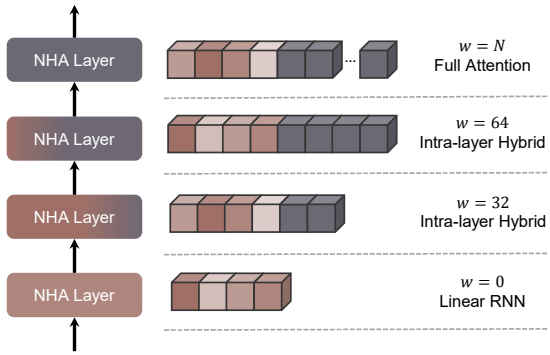

Figure 3: **Inter-Layer Hybrid in NHA**. All layers share the same NHA architecture. By varying the sliding window size w, each layer can behave as full attention ($w = N$), intra-layer hybrid ($w > 0$), or pure linear RNN ($w = 0$), enabling flexible inter-layer hybridization without changing the model structure.

The size of the sliding window can vary, ranging from 0 to the entire sequence length. This flexibility allows us to adjust the "linear ratio" of an NHA layer by modifying the window size. Setting the window size to 0 reduces the NHA to the pure linear attention model. Conversely, setting the window size to the sequence length transforms the NHA model into a Transformer with a prefix.

NHA enables inter-layer hybridization within a unified architecture. By adjusting the window size in selected layers to span the entire sequence, the model naturally realizes an inter-layer hybrid design without requiring any structural modifications.

3.4 Chunkwise Parallel Form

NHA is able to implement in a chunkwise parallel form. Let the sequence is divided into $N = \lceil T/C \rceil$ chunks of size C. For each chunk $[t]$, let the cumulative and reverse gates be $\overrightarrow{\mathcal{A}}_{[t],i} = \prod_{j=1}^{i} \mathbf{A}_{[t],j}$ and $\overleftarrow{\mathcal{A}}_{[t],i} = \prod_{j=i+1}^{C} \mathbf{A}_{[t],j}$.

Linear Memory Update We use a gated linear RNN update (Yang et al., 2023), following the associative structure of GSA (Zhang et al., 2024b):

$$\mathbf{S}_{[t]} = \mathrm{Diag}(\overrightarrow{\mathcal{A}}_{[t],C})\,\mathbf{S}_{[t-1]} + (\mathbf{K}_{[t]} \odot \overleftarrow{\mathcal{A}}_{[t]})^\top \mathbf{V}_{[t]}. \tag{6}$$

Hybrid Attention Logits. For each chunk, we compute two sets of attention logits:

1. *Linear channel logits:*

$$\mathbf{O}^k_{[t]} = \overline{\mathbf{Q}}_{[t]}\mathbf{S}_{[t-1]} + (\overline{\mathbf{Q}}_{[t]}\overline{\mathbf{K}}^\top_{[t]} \odot \mathbf{M}_{[t]})\mathbf{I}_{[t]}, \quad (7)$$

   where $\overline{\mathbf{Q}}_{[t]} = \mathbf{Q}_{[t]} \odot \overrightarrow{\mathcal{A}}_{[t]}$, $\overline{\mathbf{K}}_{[t]} = \mathbf{K}_{[t]} \odot (\overleftarrow{\mathcal{A}}_{[t]}/\overrightarrow{\mathcal{A}}_{[t],C})$, $\mathbf{I}_{[t]} = 1 - \mathbf{A}_{[t]}$ and $\mathbf{M}_{[t]}$ is the causal mask.

2. *Shifted sliding window logits:*

$$\mathbf{L}^{\text{swa}}_{[t]} = \mathbf{Q}_{[t]}\,\widetilde{\mathbf{K}}^\top_{\text{win}([t])} + \log \mathbf{M}_{\text{win}}, \quad (8)$$

   where $\text{win}([t])$ indexes positions in the current chunk and its recent W successor, $\widetilde{\mathbf{K}} = \left[\,\mathbf{0}_W;\,\mathbf{K}\,\right]_{:T}$, $\widetilde{\mathbf{V}} = \left[\,\mathbf{0}_W;\,\mathbf{V}\,\right]_{:T}$.

The two logits are concatenated for softmax:

$$\mathbf{P}^{\text{mix}}_{[t]} = \text{softmax}\big([\mathbf{O}^k_{[t]};\,\mathbf{L}^{\text{swa}}_{[t]}]\big). \quad (9)$$

The first m columns correspond to the linear channel weights $\mathbf{Q}^v_{[t]}$, and the remainder to the sliding window weights $\mathbf{P}^{\text{swa}}_{[t]}$.

Value Aggregation. We compute the outpus of linear memory branch and the sliding window branch:

$$\begin{aligned}\mathbf{O}^v_{[t]} &= \overline{\mathbf{Q}}^v_{[t]}\mathbf{S}^v_{[t-1]} + (\overline{\mathbf{Q}}^v_{[t]}\mathbf{I}^\top_{[t]} \odot \mathbf{M}_{[t]})\overline{\mathbf{V}}_{[t]},\\ \mathbf{O}^{\text{swa}}_{[t]} &= \mathbf{P}^{\text{swa}}_{[t]}\widetilde{\mathbf{V}}_{\text{win}([t])},\end{aligned} \quad (10)$$

where $\overrightarrow{\mathcal{A}}$, $\overleftarrow{\mathcal{A}}$ is absorbed into $\overline{\mathbf{Q}}^v_{[t]}$, $\overline{\mathbf{V}}_{[t]}$ the same as in Eq. 7, and $\widetilde{\mathbf{V}}$ represents the shifted $\mathbf{V}$.

The final chunkwise output is:

$$\mathbf{O}_{[t]} = \mathbf{O}^v_{[t]} + \mathbf{O}^{\text{swa}}_{[t]}. \quad (11)$$

3.5 Connections to MesaNet and Atlas

In the following, we show that our proposed NHA is closely related to recent architectures such as MesaNet (von Oswald et al., 2025) and Atlas (Behrouz et al., 2025). In particular, when equipped with matrix memory, MesaNet and Atlas can be interpreted as performing recursive least squares computations, where the models are optimized under a global L2 loss, specifically, $\mathcal{L} = \sum_{i=1}^{t}\left(\alpha_i \left\|\boldsymbol{v}_i - \boldsymbol{k}_i\boldsymbol{S}\right\|^2\right)$. In practice, the accumulation is typically confined to a limited range, thereby preserving relatively precise short-term memory while compressing long-term memory. The range aligns with the conjugate gradient step size in MesaNet and the window size in Atlas.

SWA can be viewed as a Householder-like transformation constructed from unit vectors $\boldsymbol{e}_t \in \mathbb{R}^m$. From this perspective, SWA is an unlearnable extreme case of the Delta rule, where the model retains all precise information within the window while discarding everything beyond it. NHA adopts this mechanism to preserve short-term precise memory while leveraging a linear recurrence to compress long-term memory for key and value, respectively. In contrast, the KV updates in MesaNet and Atlas are performed together, which precludes the incorporation of the softmax operation.

4 Experiments

To validate the design and capabilities of NHA, our experiments are structured to answer the following key research questions (RQs), with concise answers provided in Appendix A.

RQ1: *How does NHA perform against Transformer and other hybrid models?* (Sec. 4.2, 4.3, 4.4)

RQ2: *Can NHA achieve competitive performance to standard Transformers while offering lower cost?* (Sec. 4.5, 4.6)

RQ3: *How do NHA's hybrid components contribute to performance?* (Sec. 4.7, App. E, C.4.1)

RQ4: *Is NHA scalable for production-level LLMs?* (Sec. 4.6, App. C.3, C.5)

4.1 Experiment Setup

To answer the research questions, we conduct comprehensive experiments using a consistent setup. We pretrain all baseline hybrid models from scratch to ensure fair and consistent comparison. Following standard practice, we adopt a "one Transformer every eight layers" stacking strategy: the 340M-parameter models insert a Transformer layer as the seventh layer in every eight-layer block, while 1.3B-parameter models insert it as the first layer in every block. Comprehensive evaluation is conducted at both 340M and 1.3B scales. Additional implementation details, including dataset and training schedule, are provided in Appendix C.

4.2 Recall-Intensive Tasks

Linear models inherently face challenges in recall-intensive tasks due to their fixed-size memory states, which often leads to an performance gap

Model	Wiki. ppl↓	LMB. ppl↓	ARC_e acc	ARC_c acc_n	PIQA acc	Hella. acc_n	LMB. acc	Wino. acc	Avg. acc	FDA acc	SWDE acc	TQA acc	NQ acc	SQD acc	Drop acc	Avg. acc
340M Params 15B Tokens																
Trans++	26.88	42.15	44.91	**25.94**	64.31	34.95	32.84	51.07	42.34	46.14	25.87	45.97	18.94	33.22	20.03	31.70
GLA	28.78	39.00	44.53	22.27	63.93	34.84	32.27	51.38	41.54	11.26	16.78	43.90	12.77	27.85	17.68	21.71
GSA	28.17	42.57	45.50	24.23	64.85	35.00	30.78	50.43	41.80	6.36	16.87	42.18	14.60	21.90	16.72	19.77
GDN	26.47	**31.69**	46.04	23.55	**66.05**	35.18	34.35	50.83	42.67	20.53	23.24	44.91	14.98	28.55	16.48	24.78
Mamba2-H	25.81	43.35	**48.23**	24.49	65.23	**36.31**	30.86	48.62	42.29	55.68	39.55	45.50	18.34	27.98	17.63	34.11
GLA-H	27.94	40.12	45.20	24.32	63.93	34.42	31.83	49.96	41.61	58.86	39.36	44.14	17.01	31.37	18.21	34.83
GSA-H	26.19	46.05	45.54	23.72	65.23	35.19	30.91	50.99	41.93	**62.13**	45.36	43.78	**20.62**	31.17	18.78	36.97
GDN-H	**25.67**	51.79	45.75	23.21	65.56	35.80	29.30	51.38	41.83	52.32	38.80	39.40	15.24	29.83	17.92	32.25
NHA	25.97	38.38	47.69	24.32	65.67	36.03	32.72	**52.09**	**43.09**	53.86	**48.55**	**46.56**	20.56	**38.60**	**23.48**	**38.60**
1.3B Params 100B Tokens																
Trans++	17.61	15.86	55.01	28.07	70.08	49.21	45.60	56.27	50.71	44.32	32.43	58.47	24.49	42.59	21.56	37.31
GLA	17.61	15.38	55.18	27.56	69.86	48.89	46.05	53.91	50.24	27.61	30.93	56.28	22.27	35.04	19.45	31.93
GSA	16.69	12.62	58.33	28.33	72.25	50.98	47.43	53.43	51.79	23.25	32.80	57.05	22.96	35.57	20.65	32.05
GDN	17.14	**11.35**	56.82	27.39	71.76	49.77	49.10	51.78	51.10	30.25	27.65	58.23	23.22	34.06	20.36	32.30
Mamba2-H	**16.00**	12.77	57.74	28.24	71.93	**52.73**	48.46	55.49	52.43	65.30	50.89	55.92	28.54	41.99	22.46	44.18
GLA-H	17.58	15.72	56.73	27.56	70.40	48.71	46.71	54.46	50.76	**69.48**	46.77	57.05	27.15	40.28	21.47	43.70
GSA-H	16.22	23.09	57.79	28.07	71.71	51.70	41.92	53.35	50.76	68.30	51.83	58.59	26.42	42.36	22.42	44.99
GDN-H	16.02	11.68	58.12	27.99	70.57	52.52	**49.78**	56.27	52.54	66.76	49.02	59.06	**29.30**	41.15	24.01	44.88
NHA	16.16	12.58	**58.71**	**28.50**	**72.03**	52.08	49.19	**56.83**	**52.89**	68.30	**52.48**	**59.60**	27.18	**45.58**	**25.44**	**46.43**

Table 1: **Results on Common-Sense Reasoning Tasks and Recall-Intensive Tasks.** GDN denotes Gated DeltaNet; "-H" denotes hybrid with Transformer layers.

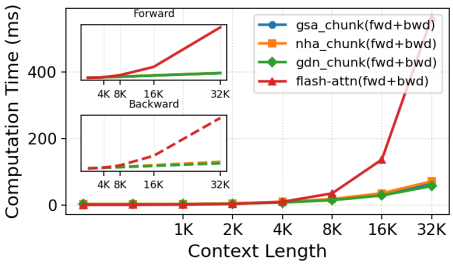

Figure 4: **Operator Training Throughput.** Throughput for forward and backward pass.

Model	NIAH-MK 1K	NIAH-MK 2K	NIAH-MK 4K	NIAH-MQ 1K	NIAH-MQ 2K	NIAH-MQ 4K	RULER-CWE 1K	RULER-CWE 2K	RULER-CWE 4K	RULER-Hotpot 2K	RULER-Hotpot 4K	RULER-Hotpot 8K
Transformer	**90.2**	**63.2**	0.4	**81.8**	64.3	13.1	42.9	20.9	0.0	21.3	2.8	0.6
Mamba2-H	13.6	3.8	4.8	52.5	55.8	**35.0**	14.5	9.0	8.5	4.4	19.4	16.0
GLA-H	46.0	23.6	12.4	37.9	30.1	25.0	50.4	12.7	10.4	26.2	20.6	19.2
GSA-H	20.0	11.2	6.4	77.7	**68.5**	34.4	42.1	24.4	**15.4**	24.8	20.6	18.0
GDN-H	59.6	29.4	15.0	56.1	41.4	27.1	**64.7**	31.0	4.1	26.5	21.6	16.8
NHA	81.4	51.8	**21.6**	62.7	50.6	28.4	63.8	**33.2**	9.3	**27.4**	**26.0**	**24.8**

Table 2: **Results on RULER tasks.** Models are trained with a context length of 2K.

when compared to Transformers. To evaluate NHA's capability in balancing long-term memory with efficient retrieval, we conducted experiments on six recall-intensive benchmarks including FDA (Arora et al., 2023), SWDE (Arora et al., 2023; Lockard et al., 2019), SQuAD (Rajpurkar et al., 2018), NQ (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017) and Drop (Dua et al., 2019). We compared NHA's performance against various pure linear models, standard Transformers, and other hybrid architectures, as summarized in Table 1. Our results demonstrate that hybrid models, in general, achieve significantly better performance than Transformer models with only a limited number of full attention layers and NHA consistently achieves the best results across these benchmarks.

4.3 Commonsense Reasoning Tasks

Commonsense reasoning benchmarks evaluate a broader and more fundamental set of capabilities, such as semantic understanding, and general world knowledge. We evaluated NHA on a suite of widely-recognized commonsense reasoning tasks, including WikiText (Merity et al., 2016), LAMBADA (Paperno et al., 2016), ARC-Easy, ARC-Challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), PiQA (Bisk et al., 2020) and Wino-Grande (Sakaguchi et al., 2019). All evaluations were performed using the lm-eval-harness framework (Gao et al., 2024). As shown in Table 1, NHA achieves the highest average score across these tasks, demonstrating that its hybrid design effectively preserves strong general reasoning abilities while enhancing efficiency.

4.4 Long-Context Benchmarks

As shown in Table 2, we evaluate models on tasks from the RULER (Hsieh et al., 2024) benchmark, including NIAH-MK, NIAH-MQ, RULER-CWE, and RULER-Hotpot. The models are trained with a context length of 2K and tested on extrapolation up to 8K. While Transformers still hold advantages on in-length needle-in-a-haystack tasks, NHA exhibits stronger extrapolation and achieves the best results across multiple tasks.

4.5 Operator Efficiency

We benchmark the forward and backward computation time of different attention operators using the `Triton-Testing-Benchmark` (Tillet et al., 2019) on one NVIDIA H100-80G GPU. For NHA, we set the long-term memory slot size to 32 and the sliding window size to 32. For GSA, we use 64 memory slots. Figure 4 reports the results for varying input sequence lengths. FlashAttention is the fastest on short sequences due to its optimized full-attention kernel, but its quadratic complexity causes computation time to grow sharply for long sequences. In contrast, NHA and GSA maintain near-linear scaling, and NHA matches GSA's speed across all lengths, as its intra-layer hybridization with a small sliding window adds negligible overhead.

4.6 Pretrained LLM Hybridization

Having established the fundamental performance and efficiency of NHA on smaller models through pretraining from scratch, we sought to further validate its practical utility and scalability. For this purpose, we conducted an experiment to structurally hybridize pre-trained open-source Transformer LLMs into NHA-based hybrids. This process involves replacing selected full-attention layers with NHA modules to obtain a faster and more efficient model without requiring full retraining.

4.6.1 Experiment Setup

We applied NHA hybridization to two pretrained LLMs: Llama-3-8B and Qwen2.5-7B, each configured with 4 full-attention layers within the hybrid architecture. This hybridization is followed by a lightweight finetuning stage, ensuring that our findings are not tied to a single architecture. Detailed initialization, layer configuration, and training procedures are provided in Appendix C.2. To further test scalability, we scale up to the Qwen3-30B-A3B model with detailed configuration in Appendix C.3.

4.6.2 Results

We evaluated the finetuned NHA-Qwen2.5-7B and NHA-Llama-3-8B against their original models, leading linear models and hybrid architectures (Table 3). NHA-Llama closely matches the original Llama and consistently offers a better trade-off between recall accuracy and computational efficiency than other efficient models. NHA-Qwen2.5 maintains strong performance on commonsense reasoning tasks. The larger gap on recall-intensive tasks may be partly due to the limited 10B-token finetuning budget, the distribution shift between the SlimPajama corpus and Qwen's pretraining data, and the hardware constraints that limited training context to 2K tokens. Both models also show a drop on MMLU, a benchmark where even state-of-the-art hybrid models underperform. Notably, despite the small finetuning budget, NHA reaches performance comparable to the strongest same-scale hybrid baselines. We further note that for existing hybrid models, the average recall accuracy tends to correlate with the number of full attention layers. Remarkably, our NHA-Llama3-8B achieves the best performance with only 4 full attention layers, highlighting its efficiency advantage (see Fig. 6).

Scaling further, hybridizing Qwen3-30B-A3B also yields competitive results across diverse benchmarks (Table 4, Appendix C.3), confirming that NHA can be applied to very large-scale models while reducing reliance on full attention layers.

4.6.3 Efficiency

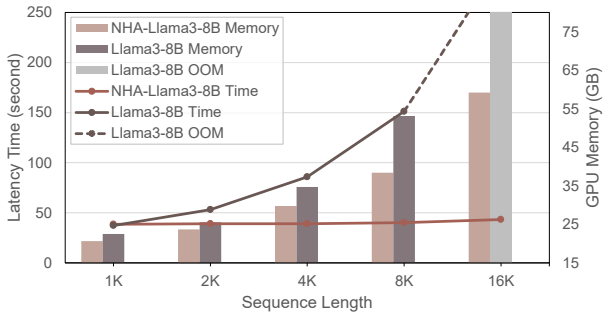

Figure 5: **Efficiency.** Inference latency and memory usage of NHA-Llama3-8B compared with Llama3-8B.

To evaluate the efficiency of NHA, we compare the inference speed and GPU memory usage of NHA-Llama3-8B with the original Llama3-8B when generating 1K tokens under varying input. As shown in Figure 5, Llama3-8B scales poorly as input length increases, whereas NHA-Llama3-8B shows much slower growth in both metrics, demonstrating superior scalability and efficiency.

Model & Scale	Training Tokens (B)	ARC_e acc	ARC_c acc_n	Hella. acc_n	LMB. acc	PIQA acc	Wino. acc	MMLU acc	Avg. acc	FDA acc	SWDE acc	SQD. acc	NQ acc	TQA. acc	Drop acc	Avg. acc
Transformer																
Mistral-7B	8000	80.85	54.01	81.08	69.49	80.90	73.64	62.50	71.78	75.30	65.32	54.01	42.76	73.16	27.55	56.35
Qwen3-8B	36000	83.42	56.74	74.90	61.11	76.61	68.11	74.93	70.83	73.48	76.85	59.89	45.01	72.63	36.61	60.75
Qwen2.5-7B	18000	80.43	51.45	78.93	64.70	78.78	73.09	74.18	71.65	82.38	75.45	54.28	48.43	77.49	38.28	62.72
Llama-3-8B	15000	80.30	53.07	79.14	68.89	79.60	72.77	65.34	71.30	82.92	72.07	54.22	43.24	74.17	33.88	60.08
Linear/Subquadratic																
Mamba-7B	1200	77.61	46.84	77.93	65.53	79.87	71.74	33.19	64.67	37.06	43.30	46.93	33.39	71.56	25.30	42.92
FalconMamba-7B	5500	83.33	58.45	80.22	61.60	80.20	75.53	60.27	71.37	48.32	59.51	49.31	37.85	75.06	30.09	50.02
RWKV-6-World-7B	1420	73.61	43.86	75.17	69.26	78.35	67.72	43.39	64.48	56.86	51.55	46.62	37.03	69.85	28.70	48.44
Griffin-7B	300	75.40	47.90	78.60	-	81.00	72.60	39.30	-	-	-	-	-	-	-	-
Hybrid																
StripedHyena-7B$_{(16)}$	-	78.62	53.75	79.11	60.78	80.03	70.32	54.07	68.10	75.30	69.82	53.81	42.41	71.86	32.34	57.59
Zamba-7B$_{(13)}$	1000	71.72	45.56	80.81	64.84	80.41	72.14	57.68	67.59	76.02	71.70	50.69	42.10	70.56	25.78	56.14
Zamba2-7B$_{(9)}$	2100	80.13	56.23	81.52	59.09	79.05	77.19	67.27	71.50	71.48	64.48	50.15	40.13	71.86	29.23	54.56
NHA-Qwen2.5-7B$_{(4)}$	10	82.74	55.97	76.49	64.54	78.73	71.03	68.80	71.19	62.58	44.61	47.26	37.47	71.98	37.47	50.23
NHA-Llama-3-8B$_{(4)}$	10	80.76	52.47	78.93	67.26	79.71	72.85	60.21	70.31	73.84	67.67	55.16	42.10	73.05	34.02	57.64

Table 3: **Comparison of Pretrained LLMs and Their NHA Hybrids.** Performance of original Transformer, linear/subquadratic, and hybrid baselines versus our NHA-hybridized Llama-3-8B and Qwen2.5-7B. Hybrid models show the number of full-attention layers in subscript parentheses.

Model	**ARC_e**	**ARC_c**	**Hella.**	**LMB.**	**PIQA**	**Wino.**	Avg.	**GPQA**	**IFVL.**	**HEval.**	**MaQA.**	**MMLU**	Avg.
Qwen3-30BA3B	78.87	**56.57**	77.63	63.05	79.49	69.61	70.87	40.62	30.58	15.85	58.09	**77.84**	44.60
NHA-Qwen3-30BA3B	**82.24**	56.40	**79.10**	**68.85**	**80.96**	**73.80**	**73.56**	**41.96**	**30.94**	**26.22**	**61.64**	75.51	**47.25**

Table 4: **Performance Comparison.** Qwen3-30B-A3B versus NHA-Qwen3-30B-A3B on standard benchmarks.

4.7 Ablation Study

In this section, we provide a detailed analysis of the components of NHA to offer insights for the design of future hybrid architectures.

4.7.1 Memory and Fusion Ablation

Variant	**Recall ↑**	**Common ↑**
NHA (full)	**38.60**	43.09
- w/o Long-Mem	29.58	40.83
- w/o Short-Mem	36.97	41.93
- w/o Token Shift	35.76	41.94
- Weighted Sum(F)	34.06	42.69
- Weighted Sum(L)	33.59	**43.12**

Table 5: **Ablation on Memory and Fusion.**

We conduct ablations to assess the contributions of NHA's key components. Removing either long-term or short-term memory leads to clear drops in performance, and eliminating token shift further degrades results by allowing overlap between the two memories. These findings highlight the complementary roles of long- and short-term memory and the necessity of keeping them distinct.

4.7.2 Alternative Fusion Strategies

To isolate the effect of the fusion mechanism, we replace NHA's unified softmax with two baselines that follow the common practice in prior "local+global" hybrids: first compute *separate* softmax attention over the long-term memory and the short-term memory, obtain their respective outputs, and then combine these outputs via a scalar weight. **Weighted Sum (F)** uses a fixed coefficient, and **Weighted Sum (L)** uses a learnable, sequence-dependent coefficient. As shown in Table 5, both alternatives underperform unified softmax, confirming the benefit of joint attention over memories.

5 Future Work

The introduction of fixed memory slots and unified long–short memory opens up several promising directions. For instance, **parameter-efficient fine-tuning (PEFT)** could be applied to learn slot initial states tailored for downstream tasks. In reasoning scenarios such as **chain-of-thought (CoT)**, reasoning chains could be selectively compressed into long-term memory, retaining essential information while reducing computational overhead.

6 Limitations

NHA achieves strong accuracy and efficiency, but it introduces additional hyperparameters such as slot size and window size that may require careful tuning to fully realize its potential. The current implementation also leaves room for further operator-level optimization, which could further improve deployment efficiency. Moreover, although NHA supports flexible adjustment of window sizes across layers, we primarily consider uniform settings, while more structured strategies such as progressive variation across depth may further enhance adaptability.

References

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *arXiv preprint arXiv:1905.13319*.

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

Simran Arora, Brandon Yang, Sabri Eyuboglu, Avanika Narayan, Andrew Hojel, Immanuel Trummer, and Christopher Ré. 2023. Language models enable simple systems for generating structured views of heterogeneous data lakes. *Preprint*, arXiv:2304.09433.

Ali Behrouz, Zeman Li, Praneeth Kacham, Majid Daliri, Yuan Deng, Peilin Zhong, Meisam Razaviyayn, and Vahab Mirrokni. 2025. Atlas: Learning to optimally memorize the context at test time. *arXiv preprint arXiv:2505.23735*.

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

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457.

Hugo Dalla-Torre, Liam Gonzalez, Javier Mendoza-Revilla, Nicolas Lopez Carranza, Adam Henryk Grzywaczewski, Francesco Oteri, Christian Dallago, Evan Trop, Bernardo P de Almeida, Hassan Sirelkhatim, and 1 others. 2025. Nucleotide transformer: building and evaluating robust foundation models for human genomics. *Nature Methods*, 22(2):287–297.

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

Xin Dong, Yonggan Fu, Shizhe Diao, Wonmin Byeon, Zijia Chen, Ameya Sunil Mahabaleshwarkar, Shih-Yang Liu, Matthijs Van Keirsbilck, Min-Hung Chen, Yoshi Suhara, and 1 others. 2024. Hymba: A hybrid-head architecture for small language models. *arXiv preprint arXiv:2411.13676*.

Jusen Du, Weigao Sun, Disen Lan, Jiaxi Hu, and Yu Cheng. 2025. Mom: Linear sequence modeling with mixture-of-memories. *arXiv preprint arXiv:2502.13685*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *arXiv preprint arXiv:1903.00161*.

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, and 5 others. 2024. A framework for few-shot language model evaluation.

Paolo Glorioso, Quentin Anthony, Yury Tokpanov, Anna Golubeva, Vasudev Shyam, James Whittington, Jonathan Pilault, and Beren Millidge. 2024a. The zamba2 suite: Technical report. *arXiv preprint arXiv:2411.15242*.

Paolo Glorioso, Quentin Anthony, Yury Tokpanov, James Whittington, Jonathan Pilault, Adam Ibrahim, and Beren Millidge. 2024b. Zamba: A Compact 7B SSM Hybrid Model. *arXiv preprint*.

Daniel Goldstein, Fares Obeid, Eric Alcaide, Guangyu Song, and Eugene Cheah. 2024. Goldfinch: High performance rwkv/transformer hybrid with linear pre-fill and extreme kv-cache compression. *arXiv preprint arXiv:2407.12077*.

Albert Gu and Tri Dao. 2024. Mamba: Linear-time sequence modeling with selective state spaces. *Preprint*, arXiv:2312.00752.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

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

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

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, and 1 others. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, and 1 others. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Disen Lan, Weigao Sun, Jiaxi Hu, Jusen Du, and Yu Cheng. 2025. Liger: Linearizing large language models to gated recurrent structures. *arXiv preprint arXiv:2503.01496*.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, Omri Abend, Raz Alon, Tomer Asida, Amir Bergman, Roman Glozman, Michael Gokhman, Avashalom Manevich, Nir Ratner, Noam Rozen, and 3 others. 2024. Jamba: A Hybrid Transformer-Mamba Language Model. *arXiv preprint*. ArXiv:2403.19887 [cs].

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, and 1 others. 2023. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130.

Colin Lockard, Prashant Shiralkar, and Xin Luna Dong. 2019. OpenCeres: When open information extraction meets the semi-structured web. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3047–3056, Minneapolis, Minnesota. Association for Computational Linguistics.

Ilya Loshchilov, Frank Hutter, and 1 others. 2017. Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*, 5.

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, and 1 others. 2025. Moba: Mixture of block attention for long-context llms. *arXiv preprint arXiv:2502.13189*.

Luke McDermott, Robert W. Heath Jr, and Rahul Parhi. 2025. LoLA: Low-Rank Linear Attention With Sparse Caching. *arXiv preprint*.

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

Tsendsuren Munkhdalai, Manaal Faruqui, and Siddharth Gopal. 2024. Leave no context behind: Efficient infinite context transformers with infini-attention. *arXiv preprint arXiv:2404.07143*, 101.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The lambada dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*.

Hao Peng, Jungo Kasai, Nikolaos Pappas, Dani Yogatama, Zhaofeng Wu, Lingpeng Kong, Roy Schwartz, and Noah A Smith. 2021. Abc: Attention with bounded-memory control. *arXiv preprint arXiv:2110.02488*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *Preprint*, arXiv:1806.03822.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Liliang Ren, Yang Liu, Yadong Lu, Yelong Shen, Chen Liang, and Weizhu Chen. 2025. Samba: Simple Hybrid State Space Models for Efficient Unlimited Context Language Modeling. *arXiv preprint*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Winogrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. 2023. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama.

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

Philippe Tillet, Hsiang-Tsung Kung, and David Cox. 2019. 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, pages 10–19.

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Johannes von Oswald, Nino Scherrer, Seijin Kobayashi, Luca Versari, Songlin Yang, Maximilian Schlegel, Kaitlin Maile, Yanick Schimpf, Oliver Sieberling, Alexander Meulemans, and 1 others. 2025. Mesanet: Sequence modeling by locally optimal test-time training. *arXiv preprint arXiv:2506.05233*.

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

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

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, YX Wei, Lean Wang, Zhiping Xiao, and 1 others. 2025. Native sparse attention: Hardware-aligned and natively trainable sparse attention. *arXiv preprint arXiv:2502.11089*.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and 1 others. 2020. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Michael Zhang, Simran Arora, Rahul Chalamala, Alan Wu, Benjamin Spector, Aaryan Singhal, Krithik Ramesh, and Christopher Ré. 2024a. Lolcats: On low-rank linearizing of large language models. *arXiv preprint arXiv:2410.10254*.

Yu Zhang, Songlin Yang, Ruijie Zhu, Yue Zhang, Leyang Cui, Yiqiao Wang, Bolun Wang, Freda Shi, Bailin Wang, Wei Bi, and 1 others. 2024b. Gated slot attention for efficient linear-time sequence modeling. *arXiv preprint arXiv:2409.07146*.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

A Answers to Research Questions

We provide concise answers to the research questions raised in Sec. 4, with supporting evidence from the corresponding sections.

RQ1: How does NHA's native hybridization design perform against Transformer and other hybrid models? NHA outperforms hybrid baselines on both recall-intensive (Sec. 4.2), commonsense reasoning tasks (Sec. 4.3) and long-context benchmarks (Sec. 4.4), confirming stronger overall effectiveness.

RQ2: Can NHA achieve competitive performance to standard Transformers while offering lower cost? Yes. NHA achieves efficiency (Sec. 4.5) and, when applied to pretrained LLMs, closely matches Transformer accuracy with reduced inference time and memory (Sec. 4.6.2, 4.6.3).

RQ3: How do NHA's hybrid components contribute, and does unified softmax improve over weighting? Ablations show both memories and token shift are essential, and unified softmax fusion clearly outperforms fixed or learned weighted fusion (Sec. 4.7). We also analyze the long-short memory (App. E) and conduct ablation study on the slot size and window size (App. C.4.1).

RQ4: Is NHA scalable for production-level LLMs? Yes. Finetuned NHA-Llama and NHA-Qwen demonstrate scalability to billion-scale LLMs with strong accuracy and efficiency gains (Sec. 4.6.2). We scale to Qwen3-30BA3B model and verify the effect of NHA (App. C.3). And NHA can utilize fewer full attention layers to get better performance than other hybrid models (App. C.5).

B Related Work and Positioning

Intra-layer hybrids. Existing intra-layer hybrid models typically combine a linear attention mechanism, which maintains long-term information in a matrix memory updated via the outer product of $\boldsymbol{k}$ and $\boldsymbol{v}$, with a local sliding window attention (SWA). Most of these approaches compute the outputs from the linear and local modules *separately* and then merge them via a fixed or learnable weighting. For example, LoL-Cats (Zhang et al., 2024a) and Liger (Lan et al., 2025) use fixed, pre-specified fusion coefficients; Infini-attention (Munkhdalai et al., 2024) employs a learnable but input-independent global coefficient; Grif-

fin (Dong et al., 2024) progressively replaces SWA layers with linear–SWA hybrids according to a predefined schedule. In contrast, NHA maintains an explicit $\boldsymbol{KV}$ cache for the local window and a compressed slot-based $\boldsymbol{KV}$ memory for the long-term context, concatenating the two and applying a *single* softmax over all keys. This yields context-dependent weights that allow the model to dynamically allocate attention between long- and short-term memory without manual fusion rules.

Inter-layer hybrids. Inter-layer hybrid architectures mix different layer types within the same model depth. Representative examples include Zamba (Glorioso et al., 2024b), Jamba (Lieber et al., 2024), Samba (Ren et al., 2025), and GoldFinch (Goldstein et al., 2024), which insert Transformer layers at fixed ratios among linear or recurrent layers. Such designs require heterogeneous module types and careful alignment of their hidden representations. By contrast, NHA uses a *single* unified layer design throughout the network. Inter-layer hybridization is achieved simply by adjusting the sliding window size per layer: setting it to zero yields a purely linear layer, while setting it to the full sequence length recovers a Transformer-like layer. This eliminates the need for architectural alignment while still allowing flexible control over the depth-wise allocation of linear and full-attention behavior.

Sharper Contrast to Prior Hybrids Many "local+global" hybrid models, such as LoL-Cats and Infini-attention, follow a two-stage design in which the long-range context is first compressed into global tokens or slots, the local and global attentions are computed independently, and the two outputs are then fused through either a fixed or a learned scalar coefficient. In these approaches, the fusion occurs at the output level, which means the trade-off between global and local information is determined outside the attention computation, is uniform across all tokens for a given query, and is unaffected by the similarity structure between the query and the keys. Gradient flow is also separated: each branch is updated independently based solely on its own attention distribution. Computationally, this design requires running two separate attention operations per query, one for the local branch and one for the global branch.

NHA differs fundamentally in both representation and computation. Long-term context is stored in the same $m \times d$ key–value slot format as local SWA tokens, enabling direct concatenation and a single softmax over all keys. This unified computation produces the global–local allocation ω_L as part of the attention distribution itself, allowing it to vary per token and per head according to similarity scores with all stored keys. As a result, gradient updates to one memory type are inherently influenced by the logits of the other, creating cross-memory coupling absent in prior designs. Theoretical differences between unified softmax and output-level fusion, including gradient coupling and context-dependent weighting, are analyzed in Sec. 3.2. In terms of efficiency, this approach requires only one attention computation over the concatenated set of keys, eliminating the duplicated cost of running separate attentions for each memory type.

Empirically, we complement this analysis with visualization. Appendix E presents heatmaps of attention scores for NHA and for representative weighted-fusion baselines, showing that NHA develops position-sensitive long/short allocation patterns that prior methods fail to capture. These visualizations provide further evidence that unified softmax learns richer and more context-dependent allocation dynamics than traditional fusion strategies.

C Experiment Details

All experiments were performed on 32 NVIDIA H100 GPUs. Training the 340M-parameter model completed in approximately 2 hours, whereas the 1.3B-parameter model required about 1 day. To ensure reproducibility, we used a fixed random seed of 42 across all training and evaluation procedures. Results are reported from a representative single run, since repeated training confirmed that performance remained consistent across runs.

C.1 Pretrain Experiment Setup

Given the scarcity of prior work directly investigating hybrid models that integrate modern linear attention mechanisms such as GLA (Yang et al., 2023), GSA (Zhang et al., 2024b), Gated DeltaNet (Yang et al., 2024), and SSMs like Mamba2 (Dao and Gu, 2024), we pretrain all baselines from scratch. Each 340M-parameter model is trained for 15B tokens, and each 1.3B-parameter model for 100B tokens on the SlimPajama (Soboleva et al., 2023) dataset. All models are optimized with AdamW (Loshchilov et al., 2017) (learning rate 3e-4, cosine schedule, weight decay 0.01, gra-

dient clipping 1.0, random seed 42).

C.2 Finetune Experiment Setup

We designed our experimental suite to cover a broad range of model sizes, datasets, and baselines, given realistic compute constraints. Our goal was to maximize coverage of scenarios most indicative of real-world performance, while ensuring reproducibility.

Parameter Inheritance Given the high parameter similarity between NHA and the standard Transformer, we initialize the corresponding $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$, and output projection weights in our NHA model directly from the pre-trained checkpoints. For the additional gating parameters introduced by NHA, we adopt a method similar to that in (Lan et al., 2025), initializing them by applying average pooling to the pretrained $\boldsymbol{K}$-projection weights. For the inter-layer hybrid configuration, we designate one layer in every eight-layer block as a full attention layer, which is achieved by setting its NHA window size to the full sequence length.

Model Hybrid Configuration For the 32-layer NHA-Llama3, we designated the first layer in every 8-layer block as a full attention layer, resulting in 4 full attention layers in total. For the 28-layer NHA-Qwen2.5, we designated the second layer in every 7-layer block as a full attention layer, also resulting in 4 full attention layers.

Training Configuration We continue the training on the SlimPajama dataset for a total of 10B tokens, divided into two distinct finetuning stages. In the first stage, we freeze the FFN parameters and exclusively finetune the attention layers for 5B tokens, allowing the model to adapt to the new hybrid attention mechanism. Subsequently, in the second stage, we apply LoRA to all model parameters and train for an additional 5B tokens to achieve efficient, full-model finetuning.

C.3 Large-Scale Hybridization with Qwen3-30B-A3B

We further scale up the hybridization experiment to the Qwen3-30B-A3B model. Considering the high computational cost of training such a large model, we adopt a conservative 2:1 hybridization strategy, and train the model on 5B tokens from the SlimPajama dataset.

On standard benchmarks including ARC-Easy, ARC-Challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), LAMBADA (Paperno et al., 2016), PIQA (Bisk et al., 2020), and WinoGrande (Sakaguchi et al., 2019), as well as broader evaluations such as GPQA (Rein et al., 2024), IFEval (Zhou et al., 2023), HumanEval (Chen et al., 2021), MathQA (Amini et al., 2019) and MMLU (Hendrycks et al., 2020). As shown in Table 4 ,the hybridized Qwen3-30B-A3B achieves competitive performance while reducing the reliance on full attention layers. These results confirm the scalability of NHA to tens-of-billions-parameter models.

m	w	Recall ↑	Common ↑
64	8	31.90	42.16
64	16	33.97	42.54
32	32	34.52	42.86
64	32	**38.60**	**43.09**
64	64	37.83	43.06

Table 6: **Ablation on Slot Size and Window Size.**

C.4 Additional Ablations

C.4.1 Ablation on Slot Size and Window Size

We jointly vary the number of long-term memory slots m and the sliding window size w to examine their combined effect on NHA's performance. All other architectural and training settings are kept identical to the main experiments. Table 6 shows that recall-intensive tasks generally benefit from larger m and moderate w, while commonsense reasoning remains stable across a wide range of configurations. Extremely small w reduces short-term precision, whereas very large w approaches full attention and erodes efficiency gains.

C.4.2 Ablation on Positional Embedding

PE Type	Recall ↑	Common ↑
RoPE (S)	**38.60**	**43.09**
RoPE (S+L)	28.46	43.01
None	26.99	41.97
Learnable (S)	30.56	41.85

Table 7: **Ablation on PE strategy.**

We evaluate four positional encoding strategies in NHA, focusing on whether and where to apply position embeddings. As shown in Table 7, "S" denotes PE applied to short-term memory and "L" to long-term memory. The results indicate that applying positional encoding solely to short-term

memory yields the best performance, as it enhances softmax attention without interfering with the implicit positional encoding already captured by the long-term memory's recursive update.

C.5 Relation Between Full Attention Layers and Recall Performance

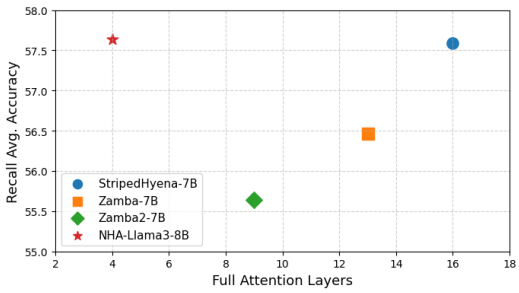

Figure 6: **Average Recall Accuracy vs. Full Attention Layers.** NHA-Llama3-8B attains the best performance with only 4 full attention layers.

D Gradient Coupling Analysis

In prior hybrids, the fusion coefficient between long- and short-term outputs is typically fixed or predicted only from the current input, without access to the similarity distribution across both memories. As a result, such coefficients cannot in general reproduce ω_L, which reflects both the current query and the aggregate interactions with all stored tokens. Because the weighting is computed jointly over all tokens in both memories, the model can express fine-grained, query-dependent trade-offs that static or input-only fusion coefficients cannot represent.

Gradient coupling. Let $Z_L = \sum_{i \in \text{long}} \exp(\boldsymbol{q}_t^\top \boldsymbol{k}_i)$ and $Z_S = \sum_{j \in \text{short}} \exp(\boldsymbol{q}_t^\top \boldsymbol{k}_j)$. Define the unified attention distribution over all keys as $p_{\text{uni}}(u) = \frac{\exp(\boldsymbol{q}_t^\top \boldsymbol{k}_u)}{Z_L + Z_S}$. Differentiating Eq. 5 with respect to the logits $\ell_u = \boldsymbol{q}_t^\top \boldsymbol{k}_u$ yields:

$$\frac{\partial \omega_L}{\partial \ell_i} = p_{\text{uni}}(i)\,(1 - \omega_L), \quad i \in \text{long}, \tag{12}$$

$$\frac{\partial \omega_L}{\partial \ell_j} = -p_{\text{uni}}(j)\,\omega_L, \quad j \in \text{short}. \tag{13}$$

These expressions show that the gradient adjusting the long/short allocation is modulated by *both* memories: changes in long-term logits depend on the total short-term mass $1 - \omega_L$, and vice versa. This creates a natural cross-memory coupling that aligns both memories in the same similarity space.

By contrast, in simple weighted fusion where the coefficient α_t is applied after computing two independent softmaxes, $\frac{\partial \alpha_t}{\partial \ell_u} = 0$ for any token u not used to compute α_t. As a result, logits in the other memory type receive no gradient signal for adjusting the allocation, leading to weaker co-adaptation between memories.

E Memory Analysis

We analyze the distribution of attention between long- and short-term memory by visualizing the average long-ratio across layers, heads, and positions (Figure 7). The results show that different layers and heads exhibit distinct preferences, indicating a specialization in memory usage. Positional patterns are also evident: early tokens before the sliding window is filled rely more on long-term memory, while later tokens increasingly attend to long-term slots for retrieving distant information.

For comparison, we examined a variant where the fusion weight between long- and short-term memory is obtained by applying a learnable linear projection to the current input (Figure 8). This variant produces a distribution that is overall more uniform across sequence positions, showing little variation between early and late tokens. In contrast, NHA exhibits a clear increase in long-term memory usage toward later positions, reflecting its ability to adapt to the growing need for retrieving distant information. This highlights that unified softmax not only enables per-token, context-dependent weighting but also captures position-sensitive patterns. This adaptive positional trend is difficult to obtain with weighted-sum fusion, where the long–short memory trade-off is externally parameterized rather than jointly learned within the attention distribution.

F Datasets

We pretrain our models on the SlimPajama dataset. For the 340M variant, we use a 15B-token sample, while the 1.3B variant is trained on a 100B-token sample. SlimPajama (Soboleva et al., 2023) is an English-language, high-quality subset of RedPajama that includes Common Crawl, Wikipedia, books, and GitHub code. It is cleaned, deduplicated, and optimized for large-scale model training.

For standard language understanding, we evaluate on the following English benchmarks: WikiText (62 test samples) (Merity et al., 2016), derived from Wikipedia articles authored by global volunteers;

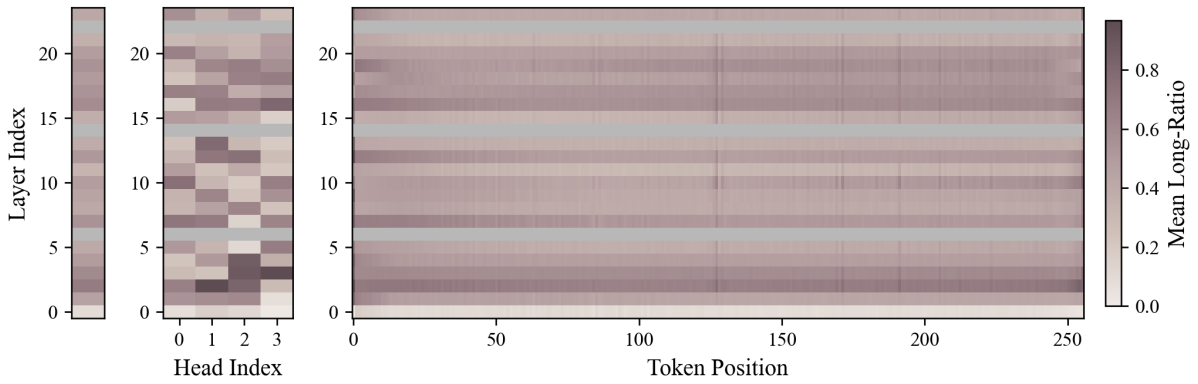

Figure 7: **Visualization of Long-Term Memory Usage.** From left to right: (1) layer-wise mean long-ratio averaged over all heads and positions; (2) average long-ratio per layer–head pair aggregated over sequence positions; (3) average long-ratio per layer–position pair aggregated over attention heads. Gray-shaded layers indicate hybrid Transformer layers in the inter-layer configuration.

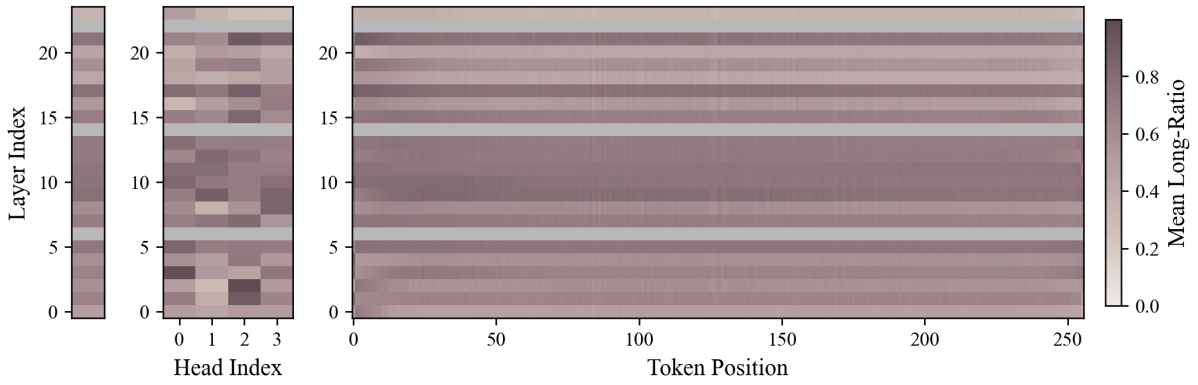

Figure 8: **Visualization of Long-Term Memory Usage.** Across layers of the input-projection fusion variant.

LAMBADA (5153) (Paperno et al., 2016), sourced from narrative books; ARC-Easy (2376) and ARC-Challenge (1172) (Clark et al., 2018), consisting of science exam questions written by educators; HellaSwag (10003) (Zellers et al., 2019), constructed from activity descriptions such as WikiHow; PiQA (3084) (Bisk et al., 2020), crowdsourced for physical commonsense reasoning; and WinoGrande (1267) (Sakaguchi et al., 2019), a large-scale pronoun resolution dataset generated through crowdsourcing.

To assess recall-intensive abilities, we adopt FDA (1102 test samples) (Arora et al., 2024, 2023), containing annotated medical device submissions; SWDE (1111) (Arora et al., 2024, 2023; Lockard et al., 2019), curated from movie and university websites; SQuAD (2984) (Rajpurkar et al., 2018), based on Wikipedia question answering; Natural Questions (3157) (Kwiatkowski et al., 2019), sourced from Google search queries; TriviaQA (1688) (Joshi et al., 2017), composed of trivia-style questions with web evidence; and DROP (2087) (Dua et al., 2019), Wikipedia passages requiring discrete reasoning.

All datasets are in English, publicly available, and released by their original creators. They are used strictly under their intended purposes and licenses, without modification or derivative dataset creation.

G Use of AI Assistants

AI assistants were employed solely for improving the clarity and readability of the manuscript through minor language polishing. They were not involved in study design, theoretical development, experiments, data analysis, or any other substantive aspect of this research.