Hardware-aligned Hierarchical Sparse Attention for Efficient Long-term Memory Access

Xiang Hu[1], **Jiaqi Leng**[2], **Jun Zhao**[2], **Kewei Tu**[3*], **Wei Wu**[1*]
[1]Ant Group, [2]Fudan University, [3]ShanghaiTech University
{aaron.hx, congyue.ww}@antgroup.com, tukw@shanghaitech.com
https://github.com/ant-research/long-context-modeling

Abstract

A key advantage of Recurrent Neural Networks (RNNs) over Transformers is their linear computational and space complexity enables faster training and inference for long sequences. However, RNNs are fundamentally unable to randomly access historical context, and simply integrating attention mechanisms may undermine their efficiency advantages. To overcome this limitation, we propose **H**ierarchical **S**parse **A**ttention (HSA), a novel attention mechanism that enhances RNNs with long-range random access flexibility while preserving their merits in efficiency and length generalization. HSA divides inputs into chunks, selects the top-k chunks and hierarchically aggregates information. The core innovation lies in learning token-to-chunk relevance based on fine-grained token-level information inside each chunk. This approach enhances the precision of chunk selection across both in-domain and out-of-domain context lengths. To make HSA efficient, we further introduce a hardware-aligned kernel design. By combining HSA with Mamba, we introduce RAMba, which achieves perfect accuracy in passkey retrieval across 64 million contexts despite pre-training on only 4K-length contexts, and significant improvements on various downstream tasks, with nearly constant memory footprint. These results show RAMba's huge potential in long-context modeling.

1 Introduction

The success of Large Language Models (LLMs) [1, 11, 59] has been largely driven by the Transformer architecture [60]. However, the quadratic computational and memory costs of self-attention make it inefficient for processing long sequences. Moreover, Transformers often struggle with inputs that exceed their pre-training length. These limitations have renewed interest in alternative architectures such as Recurrent Neural Networks (RNNs) [10, 24, 28, 33] that enable efficient, linear-time processing of sequential data while retaining a degree of extrapolation capability.

However, RNN-based models suffer from a critical limitation: the information bottleneck [58] caused by compressing variable-length contexts into fixed-dimensional representations. Unlike attention mechanisms, they lack random access to contextual information, which becomes especially problematic in tasks like passkey retrieval, where performance degrades as sequence length increases [62]. While augmenting RNNs with attention mechanisms can help mitigate this limitation, it introduces drawbacks such as poor length extrapolation, quadratic computational complexity, and substantial memory overhead during inference, ultimately undermining the original efficiency advantages of RNNs. Consequently, there remains no satisfactory RNN-based solution that can simultaneously achieve length generalization, random-access flexibility, and efficiency.

To address the trilemma, we propose a novel **H**ierarchical **S**parse **A**ttention (HSA) mechanism. Existing sparse attentions like NSA and MoBA [32, 70] typically divide sequences into chunks,

*Corresponding authors.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

allowing each token to attend to a concatenation of k selected chunks, thus can potentially achieve both efficiency and random access. While this design offers a promising step toward resolving the trilemma, a closer examination reveals a critical weakness: these methods often suffer from inaccurate chunk selection, both within the training distribution and when generalizing to longer, out-of-domain contexts. As illustrated in Figure 1(A), the issue may stem from learning token-to-chunk relevance in a chunk-unaware way, relying on token-to-token gradients instead of chunk-level feedback.

Based on this insight, HSA introduces a two-stage hierarchical mechanism for the selected chunks to enable end-to-end learning of token-to-chunk relevance. As shown in Figure 1(B), in the first stage, it applies attention separately to the tokens within each chunk to capture chunk-level information. In the second stage, it fuses the chunk-level information by applying weighted summation using token-to-chunk weights. During the backpropagation pass, token-to-chunk weights are adjusted based on the contribution of the entire chunk to the next token prediction, achieving chunk-aware learning. Experimental results show that this chunk selection mechanism remains accurate even when context lengths exceed pre-training lengths by over 10,000 times in passkey retrieval. Since each token corresponds to different k chunks, a naive implementation would require substantial memory. Thus, we further propose a hardware-aligned algorithm to achieve efficient parallel computation.

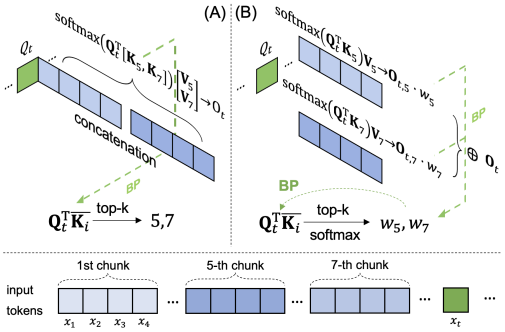

Figure 1: $\mathbf{K}_i, \mathbf{V}_i$ are the i-th chunk's key and value, with $\bar{\mathbf{K}}_i$ the mean pooling of $\mathbf{K}_i$. In (A), the chunk selection scores $\mathbf{Q}_t^\top \bar{\mathbf{K}}_i$ are learned from token-to-token interactions (*chunk-unaware*). In HSA (B), $\mathbf{Q}_t^\top \bar{\mathbf{K}}_i$ are guided by the feedback from the entire chunk (*chunk-aware*), with $\mathbf{O}_{t,i}$ the chunk-level information obtained from the i-th chunk by the t-th token.

With HSA, we propose **R**andom **A**ccess **M**am**ba** (RAMba), an extension of Mamba-2 [16] that incorporates HSA into specific layers. To maintain a constant memory footprint during inference, RAMba offloads the token-level key-value (KV) cache to CPU memory while retaining compact chunk-level representations on the GPU for efficient chunk selection. At each time step, only the KV cache of the selected chunks is loaded onto the GPU, ensuring efficient memory usage. To further minimize GPU-CPU memory swaps, RAMba leverages the hidden states derived from an intermediate layer as shared KV cache for all subsequent HSA layers, requiring just one chunk selection and swap per step. This architecture can simultaneously balance training and inference efficiency, length generalization, and long-range random access flexibility.

In our experiments, we compare RAMba with baselines such as Transformers, Mamba-2, and their variants with sliding window attention [14] and NSA [70], evaluating performance across long-range language modeling, downstream tasks, and efficiency. RAMba consistently outperforms the baselines in long-context modeling and downstream tasks while exhibiting exceptional length generalization. Notably, it is the first Mamba-based model to achieve perfect accuracy on a 64M context in the passkey retrieval task. In terms of efficiency, HSA is 3$\times$ faster than NSA and 5–25$\times$ faster than full attention for contexts of 16K tokens or more during the forward pass. Additionally, when memory offloading is enabled, RAMba maintains nearly constant memory usage. These results demonstrate RAMba's superior capability in long-text modeling. In summary, our contributions are threefold:

1. We propose HSA, a novel hierarchical attention mechanism paired with a hardware-efficient algorithm that simultaneously enables efficiency, length generalization, and flexible long-range random access.
2. Based on HSA, we introduce RAMba, which integrates the advantages of the attention mechanism into Mamba while maintaining a nearly constant memory footprint during inference.
3. We conducted comprehensive experiments on the length generalization of Mamba with various attention mechanisms. The results show that HSA excels in both performance and efficiency.

2 Related Works

Sparse Attentions. Sparse attention aims to reduce computational complexity by focusing on a limited number of tokens. For example, sliding window attention and its variants [9, 14, 41, 71] restrict computations to a fixed-size local window for each token. Such methods often sacrifice

the ability to capture long-range information. Clustering based approaches [30, 52, 61, 64] employ locality-sensitive hashing or K-Means for token clustering and perform attention with clusters. These methods often trade off efficiency for quality due to the limited accuracy of clustering. Cache eviction approaches [21, 31, 74] maintain constant memory costs by retaining only the most important tokens in the KV cache, but this limits the model's ability to access arbitrary contexts randomly. Combiner [50] utilizes a hierarchical attention mechanism by compressing tokens within chunks into single key-value pairs via max-pooling. However, this approach sacrifices the ability to capture token-level information inside the chunk, resulting in reduced accuracy. Recently, NSA [70] and MoBA [32] achieve sparsity by dividing a sequence into chunks and dynamically selecting relevant chunks for each token based on summed token-level attention scores. However, these approaches struggle with accurately identifying important chunks. Our work introduces a two-stage hierarchical attention mechanism with sparse chunk selection, achieving end-to-end learning of token-to-chunk relevance, while maintaining the ability to capture token-level information, resulting in attention flexibility, accuracy, and efficiency.

RNN-base Language Models. RNN-based language models have recently gained increased attention because their per-token inference cost remains constant as the sequence length grows. Linear Attention [28] shows that replacing softmax attention with kernel-based approximations allows Transformers to be reformulated as RNNs, achieving similar performance while benefiting from recurrent properties. Recent advancements in RNNs, such as RWKV series [42, 43], Mamba series [17, 23], GLA [66], HGRN series [46, 47], xLSTM [8] and others [67, 73], continue to push the boundaries of this approach. Our work builds on these RNN models and is in principle applicable across different RNN architectures. Meanwhile, other studies [18, 37, 51] explore hybrid architectures using RNNs for long-range memory and attention for short-range patterns. In contrast, our approach leverages RNNs to adaptively retain essential short-range information for next-token prediction, while sparse attention selectively accesses long-range context at arbitrary positions.

Context Length Extrapolation. The full attention mechanism struggles to generalize to contexts longer than those observed during pretraining [45], even with techniques like relative positional encoding [54] or attention weight scaling [63]. While methods such as Landmark Attention [35] and DCA [3] have been proposed to address this limitation, they still encounter a sharp rise in perplexity when extrapolating beyond a certain length, typically $32\times$. Recently, GCA [27] achieves perfect accuracy on 16M context lengths despite being pre-trained on only 4K lengths by integrating end-to-end retrieval within attention. However, GCA limits retrieval to once every S (usually 64) tokens, reducing its flexibility. The two-stage attention mechanism in HSA is inspired by GCA, but achieving per-token retrieval through kernel optimization.

3 Methodology

From the psychological perspective, human memory primarily comprises two main systems: working memory [6], which temporarily stores and manipulates information with limited capacity, and long-term memory [5], which stores information indefinitely with virtually unlimited capacity. Inspired by this, RAMba applies Mamba to simulate working memory by compressing variable-length contexts into a finite representation for manipulation. Meanwhile, it uses HSA to model long-term memory as an extendable KV cache, enabling efficient retrieval and attention computation. In HSA, the key innovation lies in its two-stage hierarchical attention mechanism: first, weighting over tokens within a chunk, and then weighting over chunks. By offloading KV cache to CPU memory or disk, it can theoretically maintain unlimited memory. In the following sections, we elaborate the model architecture, HSA, kernel design, and optimizations for training and inference.

3.1 Model Architecture

As shown in Figure 2(a), RAMba contains L Mamba layers, equally divided into upper and lower decoders, akin to previous works [27, 53]. Given the input sequence $\{x_1, ..., x_l\}$, their corresponding embeddings are fed into the lower decoder and the outputs are divided into $\lceil \frac{l}{S} \rceil$ chunks according to chunk size S. These chunked representations are then passed through a Transformer-based bi-directional encoder independently to form chunked memories, used for chunk selection and sparse attention in subsequent HSA layers. The upper decoder takes the outputs of the lower decoder as inputs, and alternates between one HSA layer and G Mamba layers for $\frac{L}{2G}$ times. Between the

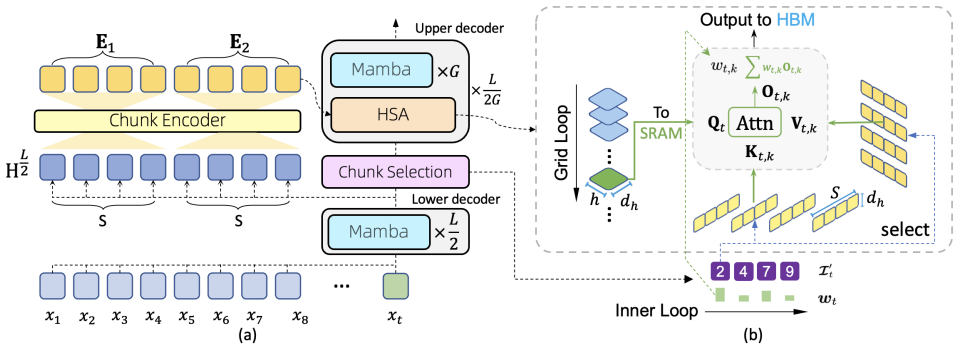

Figure 2: (a) Model architecture for RAMba. (b) Kernel design for HSA.

lower and upper decoders, a chunk selection layer selects the most relevant k chunks based on the dot-product similarity between token and chunk representations at each decoding step. HSA layers capture long-range dependencies by attending to the selected chunks.

Notations. We focus on architectures like GQA [2], where each group of h query heads shares a common KV head, and selects chunks independently. The hidden size d is divided into g groups, satisfying $g \times d_g = d$, with $d_g = h \times d_h$. Here, d_g is the dimension of each group, and d_h is the dimension per head. We denote the m-th layer output as $\mathbf{H}^m \in \mathbb{R}^{l \times d}$, with the t-step representation $\mathbf{H}_t^m \in \mathbb{R}^d$, and the encoded token representations of the i-th chunk as $\mathbf{E}_i \in \mathbb{R}^{S \times d}$. To simplify the notation, we use $\hat{\square}$ as a generic symbol to represent one of the groups, as each group behaves in the same way. For example, $\hat{\mathbf{H}}_t^m \in \mathbb{R}^{d_g}$ denotes one of the grouped representations.

Chunk Selection. At each time step t, a token uses the representation derived from the lower decoder to select top-k past chunks. Each query group independently selects K chunks, which are shared across the h heads within the group. Formally, we have:

$$\begin{aligned} \mathbf{Q}_t^{slc} &= \mathbf{W}_Q^{slc} \operatorname{norm}(\mathbf{H}_t^{\frac{L}{2}}) \\ \mathbf{K}_i^{slc} &= \mathbf{W}_K^{slc} \bar{\mathbf{E}}_i \end{aligned}, \quad \hat{s}_{t,i} = \begin{cases} \hat{\mathbf{Q}}_t^{slc\top} \hat{\mathbf{K}}_i^{slc} / \sqrt{d_g}, & i \le \lfloor \frac{t}{S} \rfloor \\ -\infty, & i > \lfloor \frac{t}{S} \rfloor \end{cases}, \quad \hat{\boldsymbol{\mathcal{I}}}_t = \{i | \operatorname{rank}(\hat{s}_{t,i}) < K\},$$

where norm is RMSNorm [72], $\mathbf{W}_Q^{slc}, \mathbf{W}_K^{slc} \in \mathbb{R}^{d \times d}$ conduct linear transformations, $\operatorname{rank}(\cdot)$ denotes the ranking position in descending order, $\bar{\mathbf{E}}_i \in \mathbb{R}^d$ is the mean-pooled representations of $\mathbf{E}_i$. $\mathbf{Q}_t^{slc}, \mathbf{K}_i^{slc} \in \mathbb{R}^d$ are representations used for chunk selection, with $\hat{\mathbf{Q}}_t^{slc}, \hat{\mathbf{K}}_i^{slc} \in \mathbb{R}^{d_g}$ as their grouped representations. For each group, $\hat{s}_{t,i}$ is the relevance score of x_t to the i-th chunk, and $\hat{\boldsymbol{\mathcal{I}}}_t$ is the selected K chunk indices for x_t.

3.1.1 Hierarchical Sparse Attention

Chunk Weights. To avoid the impact of position encoding on length generalization [29], we discard it and instead model the ordinal relationships of distances akin to the stick-breaking attention [55]. In the stick-breaking process, participants sequentially take a portion from a stick, with later participants dividing the remaining portion left by earlier ones, thus introducing a "most recent" bias. Let the total weights 1 serve as the stick and $w_{t,k}$ denote the weight assigned to the k-th chunk selected by the t-th token. For each group, we have:

$$\hat{\boldsymbol{\mathcal{I}}}_t' = \operatorname{sort}(\hat{\boldsymbol{\mathcal{I}}}_t), \quad \hat{w}_{t,k} = \hat{\beta}_{t,k} \prod_{i<k} (1 - \hat{\beta}_{t,i}) = \sigma(\hat{s}_{t,\hat{\boldsymbol{\mathcal{I}}}_{t,k}'}) \prod_{i<k} (1 - \sigma(\hat{s}_{t,\hat{\boldsymbol{\mathcal{I}}}_{t,i}'})),$$

where $\hat{\beta}_{t,k}$ is the proportion taken from the remaining attention, calculated via the sigmoid function σ. The $\operatorname{sort}(\cdot)$ function rearranges $\hat{\boldsymbol{\mathcal{I}}}_t$ in descending order to prioritize chunks closer to x_t in attention allocation. $\hat{\boldsymbol{\mathcal{I}}}_{t,k}'$ denotes the the k-th value in $\hat{\boldsymbol{\mathcal{I}}}_t'$.

Hierarchical Attention. After obtaining the weights for each chunk, x_t performs attention with tokens in each retrieved chunk separately. We denote the representation of information collected from the k-th selected chunk as $\mathbf{O}_{t,k} \in \mathbb{R}^{d_g}$. These representations are then fused using the chunk weights.

Formally, for the l-th HSA layer, we have:

$$\mathbf{Q}_t^l = \mathbf{W}_Q^l \,\mathrm{norm}(\mathbf{h}_t^{l-1}), \quad \mathbf{K}_i = \mathbf{W}_K \mathbf{E}_i, \quad \mathbf{V}_i = \mathbf{W}_V \mathbf{E}_i,$$

$$\underbrace{\hat{\mathbf{O}}_{t,k}^l = \mathrm{attn}(\hat{\mathbf{Q}}_t^l, \hat{\mathbf{K}}_{\hat{\boldsymbol{\mathcal{I}}}'_{t,k}}, \hat{\mathbf{V}}_{\hat{\boldsymbol{\mathcal{I}}}'_{t,k}})}_{\textit{token-level attention for one group}}, \quad \underbrace{\hat{\mathbf{O}}_t^l = \sum_{k<K} \hat{w}_{t,k} \hat{\mathbf{O}}_{t,k}^l}_{\textit{chunk-level attention for one group}}, \quad \underbrace{\mathbf{H}_t^{l+1} = \mathbf{H}_t^l + \mathbf{O}_t^l}_{\mathbf{O}_t^l \textit{ is the concatenation of all groups}},$$

where $\mathbf{W}_Q^l \in \mathbb{R}^{d\times d}, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d\times(g\times d_h)}$ apply linear transform to obtain query, key, and value representations, with h query head shares the same KV head. $\mathbf{K}_i, \mathbf{V}_i \in \mathbb{R}^{S\times(g\times d_h)}$ are the key and value corresponding to the i-th chunk, while $\hat{\mathbf{K}}_i, \hat{\mathbf{V}}_i \in \mathbb{R}^{S\times d_h}$ are their grouped representations, shared across all HSA layers. $\mathbf{Q}_t^l$ is the query representation of x_t at the l-th layer.

The key Difference with MoBA/NSA. The primary distinction of HSA lies in the learnable chunk retrieval module. To illustrate the difference between existing sparse attentions, such as in MoBA [32] and MiniCPM4 [56], we provide a detailed analysis with specific examples in the Appendix A.

3.1.2 Kernel Design

In HSA, each token corresponds to a distinct set of K chunks, which can lead to a substantial memory footprint in a naive implementation. Inspired by NSA, we address this issue by implementing hardware-aligned HSA kernels based on Triton [57]. As illustrated in Figure 2(b), each GPU thread loads the query representations for a single step along with the selected chunks' KV pairs for attention computation. Algorithm 1 demonstrates a parallel forward pass for one group of query heads, with multiple groups processed in parallel following the same approach. Particularly, we use softmax-off-by-one [34] to allow tokens in the current chunk to ignore any retrieved tokens. During the back-propagation process, we adopt a two-phased backward pass inspired by Dao [15]. The first stage accumulates gradients for $\mathbf{Q}$ and $\boldsymbol{w}$, followed by $\mathbf{K}$ and $\mathbf{V}$ in the second stage. The pseudo-code for this process is shown in Algorithm 2 and 3, in which $\mathbf{M}_{t,i} \in \{0,1\}^{l\times\lceil\frac{l}{S}\rceil}$ denotes whether the t-th token selects the i-th chunk, and $\mathbf{R}_{t,i} \in \mathbb{Z}^{l\times\lceil\frac{l}{S}\rceil}$ represents the index of the i-th chunk in $\boldsymbol{\mathcal{I}}'_t$.

Algorithm 1 FORWARD thread t

$\mathbf{O}'_t \leftarrow \mathbf{0}$ // *Initialize* $\mathbf{O}'_t \in \mathbb{R}^{h\times d_h}$
$\mathbf{Q}' \leftarrow$ load $\mathbf{Q}_t$ // *load* $\mathbf{Q}_t$ *to Static RAM (SRAM),* $\mathbf{Q} \in \mathbb{R}^{l\times h\times d_h}, \mathbf{Q}' \in \mathbb{R}^{h\times d_h}$
for $1 \le k \le K$ **do**
  $i \leftarrow$ load $\boldsymbol{\mathcal{I}}_{t,k}$, $w \leftarrow$ load $\boldsymbol{w}_{t,k}$ // $\boldsymbol{\mathcal{I}} \in \mathbb{Z}^{l\times K}$, $\boldsymbol{w} \in \mathbb{R}^{l\times K}$
  $\mathbf{K}' \leftarrow$ load $\mathbf{K}_i$, $\mathbf{V}' \leftarrow$ load $\mathbf{V}_i$ // $\mathbf{K}, \mathbf{V} \in \mathbb{R}^{l\times h\times d_h} \mathbf{K}', \mathbf{V}' \in \mathbb{R}^{S\times d_h}$
  $\mathbf{O}' \leftarrow \mathrm{softmax}_1(\mathbf{Q}'\mathbf{K}'^\top)\mathbf{V}'$ // *Inter-chunk token-level attention, no online softmax required.*
  $\mathbf{O}'_t \leftarrow \mathbf{O}'_t + w\mathbf{O}'$ // *Chunk-level attention via weighted sum.*
end for
$\mathbf{O}'_t \rightarrow$ write to $\mathbf{O}_t$ // *Write to High Bandwidth Memory (HBM) from Static RAM (SRAM).*

Algorithm 2 BACKWARD-$\mathbf{Q}, \boldsymbol{w}$ thread t

$\nabla\mathbf{Q}' \leftarrow \mathbf{0}, \mathbf{Q}' \leftarrow$ load $\mathbf{Q}_t, \nabla\mathbf{O}' \leftarrow$ load $\nabla\mathbf{O}_t$
for $1 \le k \le K$ **do**
  $i \leftarrow$ load $\boldsymbol{\mathcal{I}}_{t,k}$, $w \leftarrow$ load $\boldsymbol{w}_{t,k}$
  $\mathbf{K}', \mathbf{V}' \leftarrow$ load $\mathbf{K}_i, \mathbf{V}_i$ // $\mathbf{K}', \mathbf{V}' \in \mathbb{R}^{S\times d_h}$
  $\mathbf{P} \leftarrow \mathrm{softmax}_1(\mathbf{Q}'\mathbf{K}'^\top)$ // $\mathbf{P} \in \mathbb{R}^{h\times S}$
  $\mathbf{O}' \leftarrow \mathbf{P}\mathbf{V}'$ // $\mathbf{O}' \in \mathbb{R}^{h\times d_h}$
  $\mathbf{D}' \leftarrow \mathrm{rowsum}(\mathbf{O}' \circ \nabla\mathbf{O}')$ // *pointwise multiply*
  // $\mathbf{D}' \in \mathbb{R}^h$, $\nabla\mathbf{O}' \in \mathbb{R}^{h\times d_h}$
  $\mathbf{D}' \rightarrow$ write to $\mathbf{D}_{t,k}$ // $\mathbf{D} \in \mathbb{R}^{l\times K\times h}$
  $\nabla\boldsymbol{w}' \leftarrow \mathrm{rowsum}(\mathbf{D}')$ // $\nabla\boldsymbol{w}' \in \mathbb{R}$
  $\nabla\boldsymbol{w}' \rightarrow$ write to $\nabla\boldsymbol{w}_{t,k}$ // $\nabla\boldsymbol{w} \in \mathbb{R}^{l\times K}$
  $\nabla\mathbf{P} \leftarrow \nabla\mathbf{O}'\mathbf{V}'^\top$ // $\nabla\mathbf{P} \in \mathbb{R}^{h\times S}$
  $\nabla\mathbf{S} \leftarrow w\mathbf{P} \circ (\nabla\mathbf{P} - \mathbf{D}')$ // $\nabla\mathbf{S} \in \mathbb{R}^{h\times S}$
  $\nabla\mathbf{Q}_t \leftarrow \nabla\mathbf{Q}_t + \nabla\mathbf{S}\mathbf{K}'$ // $\nabla\mathbf{Q}_t \in \mathbb{R}^{h\times d_h}$
end for
$\nabla\mathbf{Q}' \rightarrow$ write to $\nabla\mathbf{Q}_t$

Algorithm 3 BACKWARD-$\mathbf{K}, \mathbf{V}$ thread i

$\mathbf{K}', \mathbf{V}' \leftarrow$ load $\mathbf{K}_i, \mathbf{V}_i$ // $\mathbf{K}', \mathbf{V}' \in \mathbb{R}^{S\times d_h}$
$\nabla\mathbf{K}', \nabla\mathbf{V}' \leftarrow \mathbf{0}$ // $\nabla\mathbf{K}', \nabla\mathbf{V}' \in \mathbb{R}^{S\times d_h}$
for $1 \le t \le l$ **do**
  if $\mathbf{M}_{t,i}$ is true **then**
    $k \leftarrow$ load $\mathbf{R}_{t,i}$, $w \leftarrow$ load $\boldsymbol{w}_{t,i}$
    $\mathbf{Q}' \leftarrow$ load $\mathbf{Q}_t$ // $\mathbf{Q}' \in \mathbb{R}^{h\times d_h}$
    $\nabla\mathbf{O}' \leftarrow$ load $\nabla\mathbf{O}_t$ // $\nabla\mathbf{O}' \in \mathbb{R}^{h\times d_h}$
    $\mathbf{D}' \leftarrow$ load $\mathbf{D}_{t,k}$ // $\mathbf{D}' \in \mathbb{R}^g$
    $\mathbf{P} \leftarrow \mathrm{softmax}_1(\mathbf{Q}'\mathbf{K}'^\top)$ // $\mathbf{P} \in \mathbb{R}^{h\times S}$
    $\nabla\mathbf{V}' \leftarrow \nabla\mathbf{V}' + w\mathbf{P}^\top\nabla\mathbf{O}'$
    $\nabla\mathbf{P} \leftarrow \nabla\mathbf{O}'\mathbf{V}'^\top$ // $\nabla\mathbf{P} \in \mathbb{R}^{h\times S}$
    $\nabla\mathbf{S} \leftarrow w\mathbf{P} \circ (\nabla\mathbf{P} - \mathbf{D}')$ // $\nabla\mathbf{S} \in \mathbb{R}^{h\times S}$
    $\nabla\mathbf{K}' \leftarrow \nabla\mathbf{K}' + \nabla\mathbf{S}^\top\mathbf{Q}'$
  end if
  $\nabla\mathbf{K}', \nabla\mathbf{V}' \rightarrow$ write to $\nabla\mathbf{K}_i, \nabla\mathbf{V}_i$
end for

3.2 Training & Inference

Training. Many efforts have been made to address Mamba's length generalization issues. A simple yet effective solution is truncated backpropagation through time (BPTT) [13, 66], in which the first state of a sequence is initialized as the final state of its preceding sequence. We follow this approach in training RAMba and other baselines. However, even though RAMba demonstrates certain extrapolation capabilities, we still observe an increase in perplexity on longer contexts. We hypothesize that RNNs' memory state provides certain shortcuts [22] for long-range attention, degrading performance on contexts largely exceeding the pre-trained length. Thus we attempt to introduce an appropriate forgetting mechanism into the memory state to disrupt the shortcuts. A straightforward way is memory reset, where sequences are divided into equal segments, and the initial state of each segment is reset to zero. To align with BPTT, we set the initial state as the last hidden state of a random segment in the previous step. In other words, for RNNs, the previous segment is randomly replaced, while the attention mechanism retains access to the original context. Our experiments show that this method effectively improves length generalization for RAMba.

Inference. A key challenge in introducing attention mechanisms to RNNs is managing the memory footprint, as the KV cache scales linearly with sequence length. Prior works [27, 35] have demonstrated the feasibility of offloading the KV cache to CPU memory and selectively loading chunks during inference. For a prompt of length L, all chunk representations are offloaded to the CPU after prefilling, while only $\mathbf{K}^{slc} \in \mathbb{R}^{\lfloor \frac{L}{S} \rfloor \times d}$ is kept in GPU memory for chunk selection. During decoding, RAMba retrieves and loads K chunks at each step for each group. Since the KV cache is shared across all HSA layers, the number of parameters exchanged between the CPU and GPU totals $g \times d_h \times K \times S$. Our efficiency analysis in § 4.4 demonstrates the overhead for memory exchange is fully acceptable. Since the memory footprint of $\mathbf{K}^{slc}$ still increases with the sequence length, to theoretically achieve constant memory, a straightforward way is to offload it to a FAISS [19] database. However, in practice, the memory footprint of $\mathbf{K}^{slc}$ is very limited, making such offloading unnecessary. Detailed analysis is elaborated in the experiments section.

4 Experiments

4.1 Setups

To ensure a fair comparison, we pre-train all 370M models from scratch with 4K context length to observe their performance and extrapolation capabilities across various tasks. For 2.7B models, the training details are presented in Appendix E.

Baselines. We adopt the Mamba-2 architecture as the backbone of the RNN model and YaRN [44] as the Transformer baseline. The parameter size of all models trained from scratch is 370M, with detailed parameters provided in Appendix B. We experiment with Mamba variants with different attention mechanisms, including sliding window attention, native sparse attention (NSA), and HSA. For sliding window attention, the window size is set to 512, incorporating two position encoding schemes: ALiBi [45] and RoPE, the latter following the settings in Samba [51]. We set the chunk size of HSA to 64 following NSA. To ensure that the field of view for sparse attention matches the sliding window size (64 * 8 = 512), we set the number of selected chunks to 8. For NSA, we use its efficient open-source implementation [*]. To isolate the effects of the sparse attention components, we disable the sliding window attention in NSA. The HSA incorporates a single-layered Transformer-based bi-directional encoder for chunk memory encoding, accounting for 5.4% of the total parameters, whose impact on fairness is minimal. HSA layers are inserted into the upper decoder every $G = 8$ Mamba layers, with other attention mechanisms like SWA and NSA following the same pattern. These settings remain consistent across all subsequent 370M models. Since the compressed attention in NSA functions similarly to Combiner [50], we do not conduct a separate comparison against Combiner. Some other related works [36] are not included in the experiments due to the lack of open-source implementations.

Pre-training. All models are pre-trained on the same 60-billion-token subset of the Pile dataset [20]. Detailed training hyper-parameters are provided in Appendix C.

[*] https://github.com/fla-org/native-sparse-attention

Models(370M)	pg19	arxiv	code	pg19	arxiv	code	pg19	arxiv	code
	eval_len=4k			eval_len=16k			eval_len=64k		
$\text{Transformer}_{\text{full\_attn}}$	18.61	4.23	3.28	539.15	199.42	62.17	$>10^4$	$>10^4$	2865.51
Mamba	17.92	4.24	3.28	17.38	3.91	3.09	17.30	3.86	**3.05**
w/ $\text{SWA}_{\text{ALiBi}}$	17.82	4.21	3.26	20.48	5.01	3.53	23.86	6.46	3.96
w/ SWA_{rope}	17.82	4.21	3.26	17.50	4.03	3.19	17.80	4.25	3.35
w/ $\text{NSA}_{\text{w/ m.r.}}$	17.87	4.20	3.25	17.31	3.87	3.06	17.31	3.87	**3.05**
w/ $\text{NSA}_{\text{w/o m.r.}}$	17.74	4.18	3.24	17.56	4.29	3.26	17.62	4.35	3.28
$\text{RAMba}_{\text{w/ m.r.}}$	17.82	4.15	3.23	17.15	**3.73**	**3.04**	**17.01**	**3.65**	3.07
$\text{RAMba}_{\text{w/o m.r.}}$	**17.63**	**4.13**	**3.21**	**17.11**	3.81	3.08	17.11	3.87	3.21
$\text{RAMba}_{\text{w/o m.r., w/o s.b.}}$	18.07	4.52	3.34	17.61	5.01	3.17	17.61	6.05	3.16

Table 1: Perplexity for long-range language modeling. We highlight the best results in **bold** and underline the second best. All models are pre-trained on 4K contexts.

Ablation Studies. We denote memory reset usage as **w/ m.r.** and its absence as **w/o m.r.**. Additionally, **w/o s.b.** refers to using softmax without positional encoding instead of stick-breaking weights. For an 8K token context, the memory is reset every 4K tokens, which aligns with the context length of other baselines. However, the chunk selection scope for sparse attention spans 8K tokens, which might be unfair to other baselines. To ensure fair comparisons, we apply the same settings to both HSA and NSA.

4.2 Long-range Langauge Modeling

Datasets. We evaluate long-range language modeling on PG19 [48], ArXiv-math [4], and Code [65].

Results. As shown in Table 1, RAMba performs better than the baselines across most datasets, on both in-domain (4K) and out-of-domain (16K & 64K) lengths. In addition, we have two findings. **First**, Mamba's perplexity decreases as the context length increases from 4K to 64K, whereas Mamba with attentions shows higher perplexity at 64K compared to 16K when not trained with memory reset. This suggests that Mamba's memory state may influence the generalization ability of attention mechanisms. **Secondly**, when trained with memory reset, Mamba with NSA or HSA all exhibit stronger length extrapolation capabilities but show a decline in in-domain performance. This result validates the effectiveness of memory reset, which essentially constrains the model to rely entirely on the content of the text for retrieval, thereby preventing the model from learning shortcuts. Since the model becomes adapted to retrieving information from longer contexts, it is reasonable that its performance declines when handling shorter contexts.

4.3 Downstream Tasks

Tasks. We evaluate various models' long-context modeling abilities on classic tasks like passkey retrieval [35] and the LongBench V2 dataset [7]. To increase task difficulty, we replace numbers in passkey retrieval with random token sequences. Since passkey retrieval is relatively simple, we further fine-tuned the models using synthetic data following RULER [25]. We use a context length of 4K for fine-tuning with a total training step size equivalent to 5% of the pre-training stage. Evaluations were conducted across different lengths on four RULER tasks: Single NIAH (S-N), Multi-queries NIAH (MQ-N), Variable Tracking (VT), and Frequent Words Extraction (FWE). To align with passkey retrieval, keys in Single NIAH were also replaced with random token sequences. We adopt a Cloze format for LongBench evaluation, following Waleffe et al. [62], to address the instruction-following challenges of small models. Since LongBench V2 is a zero-shot benchmark and thus small models may exhibit randomness, we additionally evaluate on fine-tunable datasets, including summarization tasks like XSUM [39] and CNN [38], and QA tasks like SQuaD [49], HotpotQA [68], and QuALITY [40]. NSA's implementation currently does not support generation, so its results on generative tasks are not reported.

Results. **First**, Figure 3 shows that RAMba achieves perfect accuracy in the passkey retrieval task with a 64M context, even without memory reset, while most baselines drop to nearly zero around 64K length. However, removing the chunk encoder causes RAMba's accuracy to decrease as context length grows, highlighting the encoder's importance for HSA and length generalization. One possible reason is that the hidden states are primarily optimized for predicting the next token, thus

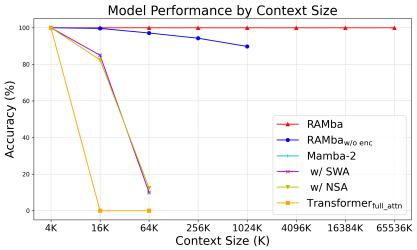

Figure 3: Passkey retrieval results.

Models(370M)	Overall	Easy	Hard	Short	Medium	Long
$\text{Transformer}_{\text{full\_attn}}$	24.8	23.4	25.7	**28.5**	19.6	**29.5**
Mamba	22.4	16.4	**26.1**	19.6	22.7	26.3
w/ $\text{SWA}_{\text{ALiBi}}$	23.3	26.9	21.0	19.6	26.8	22.1
w/ SWA_{rope}	24.2	25.7	23.2	20.3	26.3	26.3
w/ $\text{NSA}_{\text{w/ m.r.}}$	23.5	25.7	22.1	21.5	25.8	22.1
w/ $\text{NSA}_{\text{w/o m.r.}}$	22.1	26.3	19.6	17.1	26.3	22.1
$\text{RAMba}_{\text{w/ m.r.}}$	**25.7**	**28.7**	23.9	20.9	**27.8**	**29.5**
$\text{RAMba}_{\text{w/o m.r.}}$	23.5	26.3	21.7	20.9	23.7	27.4

Table 2: LongBench V2 results.

Models (370M)	S-N	MQ-N	VT	FWE	S-N	MQ-N	VT	FWE	S-N	MQ-N	VT	FWE	S-N	FWE
	ctx-len=4K				ctx-len=64K				ctx-len=256K				ctx-len=1M	
$\text{Transformer}_{\text{full\_attn}}$	95.08	**88.59**	**97.12**	60.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	—	—
Mamba-2	**96.66**	1.86	61.78	64.75	10.45	0.00	8.96	37.31	0.00	0.00	6.25	**43.75**	0.00	**40.00**
w/ SWA_{rope}	91.84	6.96	37.20	72.08	28.36	0.00	17.91	0.00	6.25	0.00	0.00	0.00	—	—
w/ NSA (w/o m.r.)	92.58	59.09	57.70	66.98	13.43	0.00	0.00	2.99	6.25	0.00	0.00	6.25	—	—
w/ NSA (w/ m.r.)	84.79	4.08	31.63	49.54	4.48	0.00	5.97	26.87	0.00	0.00	0.00	12.50	—	—
$\text{RAMba}_{\text{w/ m.r.}}$	91.74	80.61	95.18	49.81	**85.07**	**55.22**	55.22	**53.73**	**62.50**	**62.50**	**37.50**	37.50	**24.00**	20.00
$\text{RAMba}_{\text{w/o m.r.}}$	92.76	87.10	96.66	**76.81**	55.22	29.85	**68.66**	8.96	18.75	12.50	31.25	0.00	—	—
$\text{RAMba}_{\text{w/o enc}}$	92.67	81.08	57.79	75.97	41.79	23.88	14.92	26.87	6.25	12.50	0.00	6.25	—	—

Table 3: Results for selected sub-tasks in RULER. All models are pre-trained on 4K contexts.

requiring an adapter to extract information representing the current context. **Second**, LongBench's evaluation results in Table 2 are generally consistent with those of language modeling. However, since LongBench is a zero-shot benchmark, most baselines performed below the random guess rate of 25%, with the results across different sub-tasks show high variability. Thus, we believe the following SFT tasks can provide deeper insights into architectural capabilities for small models.

Table 3 presents the evaluation results of various models fine-tuned on RULER synthetic data across different context lengths. We have four interesting findings. **First, HSA demonstrates more precise chunk selection.** HSA performs comparably to full-attention on retrieval-related tasks (S-N, MQ-N, VT) over in-domain context length, while NSA lags behind both, supporting our argument that estimating chunk importance using token-to-token attention scores is inaccurate. Meanwhile, HSA achieves strong performance across retrieval tasks despite selecting chunks only once, validating the effectiveness of the hierarchical attention mechanism in learning token-to-chunk relevance. **Second, Mamba excels in sequential statistical tasks.** While Mamba-based models underperform in retrieval tasks, they outperform in Frequent Word Extraction (FWE), successfully extrapolating up to 256$\times$ the pre-training length. This advantage likely stems from Mamba's continuous memory flow mechanism, which is also partially inherited by RAMba. **Third, memory reset helps length generalization.** Models with memory reset consistently outperform those without in length extrapolation. Though performance drops significantly beyond 1M context length on more challenging retrieval tasks, it still achieves 256$\times$ extrapolation. **Fourth, templates can make a significant difference.** Single-NIAH extrapolates only to 4M, while the similar task of passkey retrieval extends to 64M, with the sole difference lying in the template of passkey retrieval being simpler. We elaborate the templates of these two tasks in Appendix D. It suggests that a simpler pattern may facilitate learning more generalizable patterns. Still, precise chunk selection for extremely long contexts remains an open challenge for future work.

Models (370M)	$\text{XSUM}_{\text{R-1/R-2/R-L}}$ avg-len=498	$\text{CNN}_{\text{R-1/R-2/R-L}}$ avg-len=883	$\text{SQuaD}_{\text{EM/F1}}$ avg-len=174	$\text{HotpotQA}_{\text{EM/F1}}$ avg-len=1428	$\text{QuALITY}_{\text{Acc}}$ avg-len=7745	AVG.
Mamba-2	30.40/11.89/24.53	37.97/17.54/35.84	41.33/52.03	18.70/26.20	33.32	31.03
w/ $\text{SWA}_{\text{ALiBi}}$	30.78/12.11/24.79	38.96/17.84/36.73	50.54/61.23	19.96/27.58	29.48	32.57
w/ SWA_{rope}	**30.88/12.33/24.99**	38.78/17.67/36.57	**51.46/61.91**	19.89/28.00	32.89	32.61
w/ NSA	—	—	—	—	31.40	—
$\text{Transformer}_{\text{full\_attn}}$	30.10/11.63/24.23	38.25/17.89/36.14	45.50/56.13	21.90/29.49	33.46	32.82
RAMba	30.81/12.25/24.80	**39.11/18.04/36.85**	48.24/59.17	**22.30/30.53**	**34.13**	**33.64**

Table 4: Downstream task evaluations.

Table 4 shows the results of the model after fine-tuning on summarization and generation tasks. RAMba still outperforms in the vast majority of tasks, but two observations are worth noting. **First**, Mamba with SWA demonstrates advantages over other models on the SQuaD task. One potential reason is that the SQuaD dataset contains shorter texts, even less than the sliding window, making

SWA able to randomly access all contexts. This phenomenon shows that integrating random access can improve RNN performance even for short contexts. **Second**, despite lagging behind Mamba in perplexity (PPL), the Transformer demonstrates strong downstream performance, ranking just behind RAMba. This further validates the importance of random access capability for downstream tasks.

Models	#params.	LongBench						RULER$_{\text{S-N/MQ-N/VT/FWE}}$		
		Overall	Easy	Hard	Short	Medium	Long	ctx-len=4K	ctx-len=64K	ctx-len=1M
Mamba-2	2.7B	26.8	23.4	29.0	29.7	27.3	21.1	89.24/26.62/24.68/**77.09**	0.00/0.00/0.00/0.00	0.00/0.00/0.00/0.00
RAMba$_{\text{w/ m.r.}}$	2.7B + 110M	**27.3**	25.1	28.6	30.4	27.3	22.1	**91.00/86.36/96.47**/42.95	**83.58/76.12/100.00/28.36**	**25.00/13.33/50.00/37.14**

Table 5: Results of 2.7B models trained on 4K contexts.

Table 5 shows the results of the 2.7B models. These results show that even after scaling up, RAMba still maintains a significant lead. Details about training can be found in Appendix E.

4.4 Efficiency Analysis

Experimental Setup. To evaluate the scalability and efficiency of HSA, we measure the runtime and memory footprint of FlashAttention-2, NSA, and HSA operators across different sequence lengths, the throughput of various models at different scales, and the per-token time consumption during inference. Runtime measurements were conducted with three attention layers only, excluding additional components such as MLPs or Mamba. In HSA, a single chunk selection is shared across all three layers, while SWA remains disabled in NSA. The memory footprint is reported as a ratio of the memory occupied by Mamba 370M. We use **w/ offloading** and **w/o offloading** to denote whether memory offloading is enabled. To further evaluate the benefits of shared chunk selection, we introduce an ablation group (**w/o sharing**), where chunk selection and CPU-GPU memory exchanges happen in each HSA layer. When measuring training throughput, we enable FSDP [75] and gradient checkpointing [12], running models on 16 $\times$ Physics Processing Units (PPUs), each with approximately half the computational power of an A100 GPU.

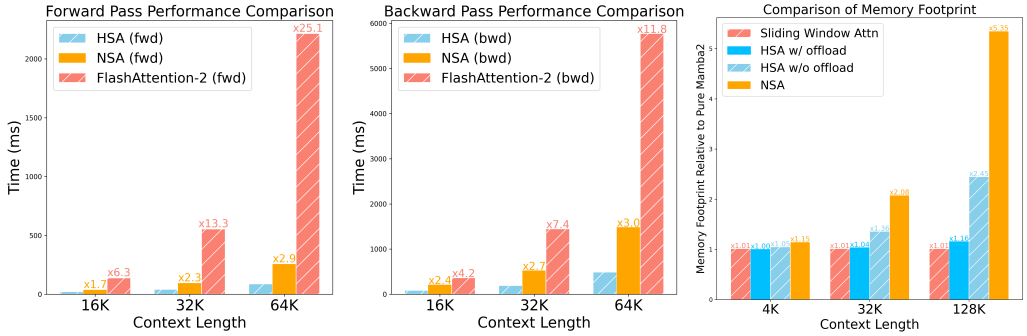

Figure 4: Comparison of attention computation time: 3 attention layers per group. (The lower the better)

Models	Parameter Size			
	370M↑	780M↑	1.4B↑	3B↑
Mamba	16.44	9.71	6.53	3.46
w/ SWA	15.64	9.24	6.27	3.31
w/ full_attn	10.00	5.73	4.31	2.07
w/ NSA	14.35	8.70	5.46	3.18
RAMba	14.80	8.76	5.70	3.18

Table 6: Training throughput (10^3 tokens/s) with context length=32K.

Models	Prompt-Length		
	4K↓	16K↓	64K↓
Transformer$_{\text{full_attn}}$	2.26	8.90	32.12
Mamba-2	2.92	2.82	2.84
RAMba$_{\text{w/o offloading}}$	3.14	3.05	2.76
RAMba$_{\text{w/ offloading}}$	3.97	4.19	4.02
RAMba$_{\text{w/ offloading, w/o sharing}}$	5.98	5.87	5.95

Table 7: Inference time cost (seconds, prefilling time excluded) for generating 100 tokens (batch-size=16)

Results. Figure 4 compares the time consumption and memory footprint of three operators across different context lengths. Both NSA and HSA, as sparse attention mechanisms, significantly outperform Flash-Attention in terms of speed. HSA is faster than NSA because it performs chunk selection—the only operation with quadratic complexity—only once, and shares it across all HSA layers. In contrast, NSA involves computations with quadratic complexity like compressed token attention and chunk selection in every layer, which increases its time consumption. For memory footprint, enabling memory offloading drastically reduces GPU memory usage and slows its growth with

increasing context length. Despite introducing CPU-GPU memory exchange, the impact on inference speed is limited, as reported in Table 7. Table 6 shows that RAMba achieves 90% of Mamba's training throughput. Although HSA is faster than NSA, the additional encoder sometimes offsets this advantage at certain scales. Nonetheless, the results clearly highlight HSA's high efficiency, excellent scalability during training, and near-constant memory usage during inference.

5 Conclusion

In this work, we present Hierarchical Sparse Attention (HSA) and build on it to propose RAMba, which integrates an RNN backbone, length-generalizable sparse attention, and a simple forgetting mechanism. This architecture strikes a strong balance between performance, efficiency, long-range access, and length generalization, offering a foundation for language models with permanent memory.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://openreview.net/forum?id=hmOwOZWzYE.

[3] Chenxin An, Fei Huang, Jun Zhang, Shansan Gong, Xipeng Qiu, Chang Zhou, and Lingpeng Kong. Training-free long-context scaling of large language models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=If4xW9vF7U.

[4] Zhangir Azerbayev, Edward Ayers, and Bartosz Piotrowski. Proof-pile: A pre-training dataset of mathematical text, 2023. URL https://huggingface.co/datasets/hoskinson-center/proof-pile.

[5] Alan Baddeley. The episodic buffer: a new component of working memory? *Trends in Cognitive Sciences*, 4(11):417–423, 2000. ISSN 1364-6613. doi: https://doi.org/10.1016/S1364-6613(00)01538-2. URL https://www.sciencedirect.com/science/article/pii/S1364661300015382.

[6] Alan D. Baddeley and Graham Hitch. Working memory. volume 8 of *Psychology of Learning and Motivation*, pages 47–89. Academic Press, 1974. doi: https://doi.org/10.1016/S0079-7421(08)60452-1. URL https://www.sciencedirect.com/science/article/pii/S0079742108604521.

[7] Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks, 2025. URL https://arxiv.org/abs/2412.15204.

[8] Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xlstm: Extended long short-term memory, 2024. URL https://arxiv.org/abs/2405.04517.

[9] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *CoRR*, abs/2004.05150, 2020. URL https://arxiv.org/abs/2004.05150.

[10] Yoshua Bengio, Patrice Y. Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Networks*, 5(2):157–166, 1994. doi: 10.1109/72.279181. URL https://doi.org/10.1109/72.279181.

[11] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

[12] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *CoRR*, abs/1604.06174, 2016. URL http://arxiv.org/abs/1604.06174.

[13] Yingfa Chen, Xinrong Zhang, Shengding Hu, Xu Han, Zhiyuan Liu, and Maosong Sun. Stuffed mamba: State collapse and state capacity of rnn-based long-context modeling. *CoRR*, abs/2410.07145, 2024. doi: 10.48550/ARXIV.2410.07145. URL https://doi.org/10.48550/arXiv.2410.07145.

[14] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *CoRR*, abs/1904.10509, 2019. URL http://arxiv.org/abs/1904.10509.

[15] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=mZn2Xyh9Ec.

[16] Tri Dao and Albert Gu. Transformers are ssms: generalized models and efficient algorithms through structured state space duality. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[17] Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=ztn8FCR1td.

[18] Soham De, Samuel L. Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, Guillaume Desjardins, Arnaud Doucet, David Budden, Yee Whye Teh, Razvan Pascanu, Nando De Freitas, and Caglar Gulcehre. Griffin: Mixing gated linear recurrences with local attention for efficient language models, 2024. URL https://arxiv.org/abs/2402.19427.

[19] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. *CoRR*, abs/2401.08281, 2024. doi: 10.48550/ARXIV.2401.08281. URL https://doi.org/10.48550/arXiv.2401.08281.

[20] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The pile: An 800gb dataset of diverse text for language modeling, 2020. URL https://arxiv.org/abs/2101.00027.

[21] Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive KV cache compression for llms. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=uNrFpDPMyo.

[22] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard S. Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. Shortcut learning in deep neural networks. *Nat. Mach. Intell.*, 2(11):665–673, 2020. doi: 10.1038/S42256-020-00257-Z. URL https://doi.org/10.1038/s42256-020-00257-z.

[23] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *CoRR*, abs/2312.00752, 2023. doi: 10.48550/ARXIV.2312.00752. URL https://doi.org/10.48550/arXiv.2312.00752.

[24] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

[25] Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=kIoBbc76Sy.

[26] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

[27] Xiang Hu, Zhihao Teng, Jun Zhao, Wei Wu, and Kewei Tu. Efficient length-generalizable attention via causal retrieval for long-context language modeling, 2025. URL https://arxiv.org/abs/2410.01651.

[28] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 5156–5165. PMLR, 2020. URL http://proceedings.mlr.press/v119/katharopoulos20a.html.

[29] Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. The impact of positional encoding on length generalization in transformers. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/4e85362c02172c0c6567ce593122d31c-Abstract-Conference.html.

[30] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=rkgNKkHtvB.

[31] Chao Lou, Zixia Jia, Zilong Zheng, and Kewei Tu. Sparser is faster and less is more: Efficient sparse attention for long-range transformers. *CoRR*, abs/2406.16747, 2024. doi: 10.48550/ARXIV.2406.16747. URL https://doi.org/10.48550/arXiv.2406.16747.

[32] Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, Zhiqi Huang, Huan Yuan, Suting Xu, Xinran Xu, Guokun Lai, Yanru Chen, Huabin Zheng, Junjie Yan, Jianlin Su, Yuxin Wu, Neo Y. Zhang, Zhilin Yang, Xinyu Zhou, Mingxing Zhang, and Jiezhong Qiu. Moba: Mixture of block attention for long-context llms. *CoRR*, abs/2502.13189, 2025. doi: 10.48550/ARXIV.2502.13189. URL https://doi.org/10.48550/arXiv.2502.13189.

[33] Eric Martin and Chris Cundy. Parallelizing linear recurrent neural nets over sequence length. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HyUNwulC-.

[34] Evan Miller. Attention is off by one, 2023. URL https://www.evanmiller.org/attention-is-off-by-one.html.

[35] Amirkeivan Mohtashami and Martin Jaggi. Random-access infinite context length for transformers. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/ab05dc8bf36a9f66edbff6992ec86f56-Abstract-Conference.html.

[36] Tsendsuren Munkhdalai, Manaal Faruqui, and Siddharth Gopal. Leave no context behind: Efficient infinite context transformers with infini-attention. *CoRR*, abs/2404.07143, 2024. doi: 10.48550/ARXIV.2404.07143. URL https://doi.org/10.48550/arXiv.2404.07143.

[37] Tsendsuren Munkhdalai, Manaal Faruqui, and Siddharth Gopal. Leave no context behind: Efficient infinite context transformers with infini-attention, 2024. URL https://arxiv.org/abs/2404.07143.

[38] Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence rnns and beyond. In Yoav Goldberg and Stefan Riezler, editors, *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 280–290. ACL, 2016. doi: 10.18653/V1/K16-1028. URL https://doi.org/10.18653/v1/k16-1028.

[39] Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 1797–1807. Association for Computational Linguistics, 2018. doi: 10.18653/V1/D18-1206. URL https://doi.org/10.18653/v1/d18-1206.

[40] Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, and Samuel R. Bowman. Quality: Question answering with long input texts, yes! In Marine Carpuat, Marie-Catherine de Marneffe, and Iván Vladimir Meza Ruíz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 5336–5358. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.NAACL-MAIN.391. URL https://doi.org/10.18653/v1/2022.naacl-main.391.

[41] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 4052–4061. PMLR, 2018. URL http://proceedings.mlr.press/v80/parmar18a.html.

[42] Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Leon Derczynski, Xingjian Du, Matteo Grella, Kranthi Kiran GV, Xuzheng He, Haowen Hou, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartlomiej Koptyra, Hayden Lau, Jiaju Lin, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Guangyu Song, Xiangru Tang, Johan S. Wind, Stanislaw Wozniak, Zhenyuan Zhang, Qinghua Zhou, Jian Zhu, and Rui-Jie Zhu. RWKV: reinventing rnns

for the transformer era. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 14048–14077. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.FINDINGS-EMNLP.936. URL https://doi.org/10.18653/v1/2023.findings-emnlp.936.

[43] Bo Peng, Daniel Goldstein, Quentin Anthony, Alon Albalak, Eric Alcaide, Stella Biderman, Eugene Cheah, Xingjian Du, Teddy Ferdinan, Haowen Hou, Przemyslaw Kazienko, Kranthi Kiran GV, Jan Kocon, Bartlomiej Koptyra, Satyapriya Krishna, Ronald McClelland Jr., Niklas Muennighoff, Fares Obeid, Atsushi Saito, Guangyu Song, Haoqin Tu, Stanislaw Wozniak, Ruichong Zhang, Bingchen Zhao, Qihang Zhao, Peng Zhou, Jian Zhu, and Rui-Jie Zhu. Eagle and finch: RWKV with matrix-valued states and dynamic recurrence. *CoRR*, abs/2404.05892, 2024. doi: 10.48550/ARXIV.2404.05892. URL https://doi.org/10.48550/arXiv.2404.05892.

[44] Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient context window extension of large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=wHBfxhZu1u.

[45] Ofir Press, Noah A. Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=R8sQPpGCv0.

[46] Zhen Qin, Songlin Yang, and Yiran Zhong. Hierarchically gated recurrent neural network for sequence modeling. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=P1TCHxJwLB.

[47] Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. HGRN2: Gated linear RNNs with state expansion. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=y6SqbJfCSk.

[48] Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SylKikSYDH.

[49] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. In Jian Su, Xavier Carreras, and Kevin Duh, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392. The Association for Computational Linguistics, 2016. doi: 10.18653/V1/D16-1264. URL https://doi.org/10.18653/v1/d16-1264.

[50] Hongyu Ren, Hanjun Dai, Zihang Dai, Mengjiao Yang, Jure Leskovec, Dale Schuurmans, and Bo Dai. Combiner: Full attention transformer with sparse computation cost. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 22470–22482, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/bd4a6d0563e0604510989eb8f9ff71f5-Abstract.html.

[51] Liliang Ren, Yang Liu, Yadong Lu, yelong shen, Chen Liang, and Weizhu Chen. Samba: Simple hybrid state space models for efficient unlimited context language modeling. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=bIlnpVM4bc.

[52] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Trans. Assoc. Comput. Linguistics*, 9:53–68, 2021. doi: 10.1162/TACL_A_00353. URL https://doi.org/10.1162/tacl_a_00353.

[53] Ohad Rubin and Jonathan Berant. Retrieval-pretrained transformer: Long-range language modeling with self-retrieval. *Transactions of the Association for Computational Linguistics*, 12:1197–1213, 2024. doi: 10.1162/tacl_a_00693. URL https://aclanthology.org/2024.tacl-1.66/.

[54] Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *CoRR*, abs/2104.09864, 2021. URL https://arxiv.org/abs/2104.09864.

[55] Shawn Tan, Songlin Yang, Aaron Courville, Rameswar Panda, and Yikang Shen. Scaling stick-breaking attention: An efficient implementation and in-depth study. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=r8J3DSD5kF.

[56] MiniCPM Team, Chaojun Xiao, Yuxuan Li, Xu Han, Yuzhuo Bai, Jie Cai, Haotian Chen, Wentong Chen, Xin Cong, Ganqu Cui, Ning Ding, Shengda Fan, Yewei Fang, Zixuan Fu, Wenyu Guan, Yitong Guan, Junshao Guo, Yufeng Han, Bingxiang He, Yuxiang Huang, Baoxi Ji, Cunliang Kong, Qiuzuo Li, Siyuan Li, Wenhao Li, Xin Li, Yanghao Li, Yishan Li, Zhen Li, Dan Liu, Biyuan Lin, Yankai Lin, Xiang Long, Quanyu Lu, Yaxi Lu, Peiyan Luo, Hongya Lyu, Litu Ou, Yinxu Pan, Lushi Pu, Zekai Qu, Qundong Shi, Zijun Song, Jiayuan Su, Zhou Su, Ao Sun, Xianghui Sun, Peijun Tang, Fangzheng Wang, Feng Wang, Shuo Wang, Yudong Wang, Zheng Wang, Yesai Wu, Zhenyu Xiao, Jie Xie, Zihao Xie, Xiaoyue Xu, Yukun Yan, Jiarui Yuan, Jinqian Zhang, Kaihuo Zhang, Lei Zhang, Linyue Zhang, Xueren Zhang, Yudi Zhang, Hengyu Zhao, Weilin Zhao, Weilun Zhao, Yuanqian Zhao, Zhi Zheng, Chuyue Zhou, Ge Zhou, Jie Zhou, Wei Zhou, Yanghao Zhou, Zihan Zhou, Zixuan Zhou, Zhiyuan Liu, Guoyang Zeng, Chao Jia, Dahai Li, and Maosong Sun. Minicpm4: Ultra-efficient llms on end devices, 2025.

[57] Philippe Tillet, Hsiang-Tsung Kung, and David D. Cox. Triton: an intermediate language and compiler for tiled neural network computations. In Tim Mattson, Abdullah Muzahid, and Armando Solar-Lezama, editors, *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages, MAPL@PLDI 2019, Phoenix, AZ, USA, June 22, 2019*, pages 10–19. ACM, 2019. doi: 10.1145/3315508.3329973. URL https://doi.org/10.1145/3315508.3329973.

[58] Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method, 2000. URL https://arxiv.org/abs/physics/0004057.

[59] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[60] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[61] Apoorv Vyas, Angelos Katharopoulos, and François Fleuret. Fast transformers with clustered attention. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/f6a8dd1c954c8506aadc764cc32b895e-Abstract.html.

[62] Roger Waleffe, Wonmin Byeon, Duncan Riach, Brandon Norick, Vijay Korthikanti, Tri Dao, Albert Gu, Ali Hatamizadeh, Sudhakar Singh, Deepak Narayanan, Garvit Kulshreshtha, Vartika Singh, Jared Casper, Jan Kautz, Mohammad Shoeybi, and Bryan Catanzaro. An empirical study of mamba-based language models, 2024. URL https://arxiv.org/abs/2406.07887.

[63] Jie Wang, Tao Ji, Yuanbin Wu, Hang Yan, Tao Gui, Qi Zhang, Xuanjing Huang, and Xiaoling Wang. Length generalization of causal transformers without position encoding. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14024–14040, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.834. URL https://aclanthology.org/2024.findings-acl.834/.

[64] Shuohang Wang, Luowei Zhou, Zhe Gan, Yen-Chun Chen, Yuwei Fang, Siqi Sun, Yu Cheng, and Jingjing Liu. Cluster-former: Clustering-based sparse transformer for long-range dependency encoding. *CoRR*, abs/2009.06097, 2020. URL https://arxiv.org/abs/2009.06097.

[65] Thomas Wolf, Loubna Ben Allal, Leandro von Werra, Li Jia, and Armel Zebaze. A dataset of python files from github, 2023. URL https://github.com/huggingface/blog/blob/main/codeparrot.md.

[66] Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=ia5XvxFUJT.

[67] Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length, 2025. URL https://arxiv.org/abs/2406.06484.

[68] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018*

Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018, pages 2369–2380. Association for Computational Linguistics, 2018. doi: 10.18653/V1/D18-1259. URL https://doi.org/10.18653/v1/d18-1259.

[69] Howard Yen, Tianyu Gao, and Danqi Chen. Long-context language modeling with parallel context encoding. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2588–2610, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.142. URL https://aclanthology.org/2024.acl-long.142/.

[70] Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Y. X. Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention. *CoRR*, abs/2502.11089, 2025. doi: 10.48550/ARXIV.2502.11089. URL https://doi.org/10.48550/arXiv.2502.11089.

[71] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/c8512d142a2d849725f31a9a7a361ab9-Abstract.html.

[72] Biao Zhang and Rico Sennrich. *Root mean square layer normalization*. Curran Associates Inc., Red Hook, NY, USA, 2019.

[73] Yu Zhang, Songlin Yang, Rui-Jie Zhu, Yue Zhang, Leyang Cui, Yiqiao Wang, Bolun Wang, Freda Shi, Bailin Wang, Wei Bi, Peng Zhou, and Guohong Fu. Gated slot attention for efficient linear-time sequence modeling. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=jY4PhQibmg.

[74] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark W. Barrett, Zhangyang Wang, and Beidi Chen. H2O: heavy-hitter oracle for efficient generative inference of large language models. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/6ceefa7b15572587b78ecfcebb2827f8-Abstract-Conference.html.

[75] Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. Pytorch fsdp: Experiences on scaling fully sharded data parallel, 2023. URL https://arxiv.org/abs/2304.11277.

Limitations

Although the method proposed in this work is theoretically applicable to all RNNs, the experimental section of this work mainly focuses on Mamba.

Considering computational resources, this work does not discuss the performance of models larger than 3B parameters.

A How HSA achieves accurate chunk retrieval

The core principle of previous sparse attentions' chunk selection lies in approximating the token-to-chunk relevance using unnormalized attention scores. In self-attention, the relevance of token j to token i is defined as:

$$p_{i,j} = \frac{e^{\mathbf{logits}_{i,j}}}{\mathbf{Z}_i}, \mathbf{Z}_i = \sum_{j<i} e^{\mathbf{logits}_{i,j}}, \tag{1}$$

where $\mathbf{logits}_{i,j} = \mathbf{q}_i^\top \mathbf{k}_j$ is the dot product between the query $\mathbf{q}_i$ of token i and the key $\mathbf{k}_j$ of token j.

The relevance of chunk c to token i is ideally the sum of the relevance of all tokens within that chunk:

$$r_{i,c} = \sum_{j \in \mathcal{C}} p_{i,j} = \frac{1}{\mathbf{Z}_i} \sum_{j \in \mathcal{C}} e^{\mathbf{logits}_{i,j}}, \tag{2}$$

where $\mathcal{C}$ denotes tokens in chunk c. However, calculating $r_{i,c}$ requires computing $\mathbf{Z}_i$ (the full softmax normalization across all tokens), which would necessitate full computation and thus undermine the computational efficiency.

To ensure efficiency, they instead approximate chunk relevance using the mean-pooled representation of keys:

$$r'_{i,c} = \mathbf{q}_i^\top \mathbf{K}_c = \mathbf{q}_i^\top \frac{1}{S} \sum_{j \in \mathcal{C}} \mathbf{k}_j = \frac{1}{S} \sum_{j \in \mathcal{C}} \mathbf{q}_i^\top \mathbf{k}_j = \frac{1}{S} \sum_{j \in \mathcal{C}} \mathbf{logits}_{i,j}, \tag{3}$$

where $\mathbf{K}_c$ represents the mean-pooling of key representations within the c-th chunk. While this approximation bypasses the need for normalized softmax scores across all tokens, it introduces a discrepancy between $r'_{i,c}$ (the approximation) and $r_{i,c}$ (the ideal one).

Let's consider the following example shown in Figure 5, assuming every 2 tokens form a chunk. If

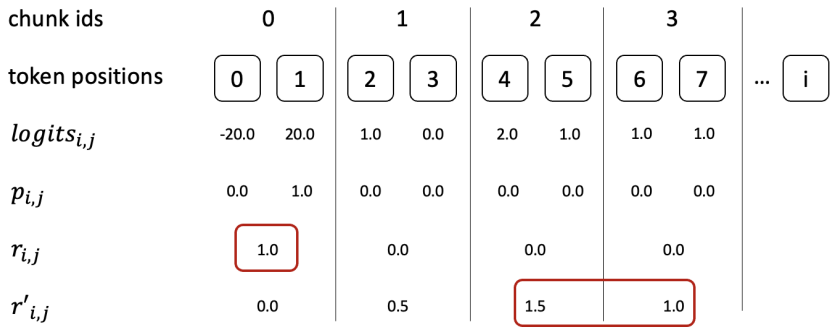

Figure 5: How unnormalized scores mislead the chunk selection.

only the top-2 chunks can be selected, they would choose chunks 2 and 3 according to $r'_{i,c}$, thus missing the chunk 0 with the highest sum of attention weights.

Assuming they select the top-2 chunks for sparse attention, the subsequent attention process is shown in Figure 6. Throughout this process, $r'_{i,c}$ only participates in chunk selection but not in the forward computation, nor does it receive gradients. The inaccurate chunk selection issue stems from using unnormalized attention logits to estimate chunk importance, with $r'_{i,c}$ not learnable, making the inaccuracy of chunk selection unavoidable.

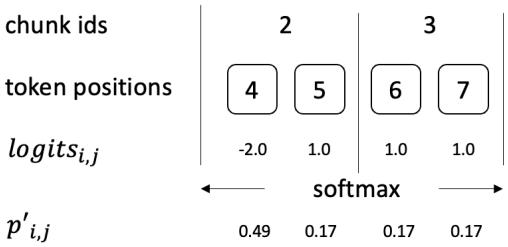

Figure 6: Applying attention over the concatenation of selected chunks.

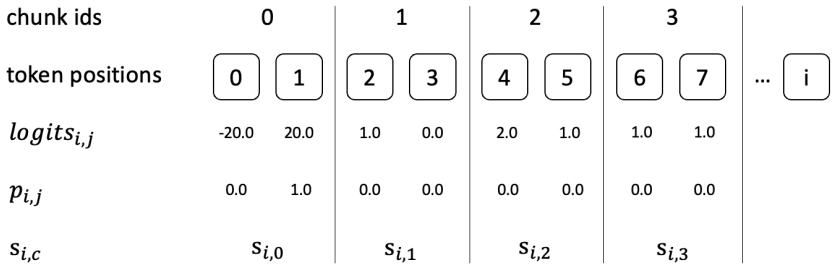

Figure 7: The chunk selection stage of HSA

For the same example, HSA works as shown in Figure 7 where $s_{i,c}$ represents learnable token-to-chunk relevance scores. HSA selects chunks 2 and 3 according to $s_{i,c}$, and applies the hierarchical attention over selected chunks as shown in Figure 8. In this process, $w_{i,c}$, derived from $s_{i,c}$, par-

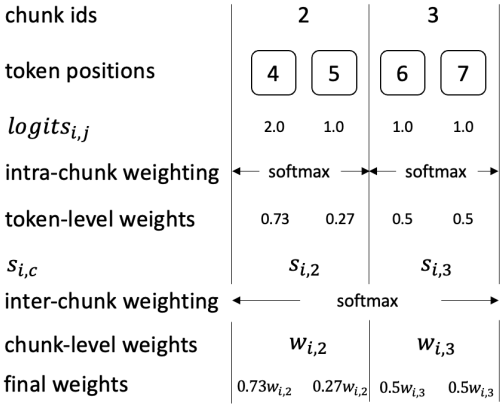

Figure 8: Applying hierarchical attention over selected chunks.

ticipates in the final attention weight computation and the whole forward pass. This allows it to receive gradients and allocate higher weights to chunks with more important tokens. Even if random initialization initially misses the most relevant chunk 0, continuous training enables the model to gradually learn to select the most relevant chunks.

B Hyper-parameters

Architecture	Transformer	Mamba	Mamba+NSA	Mamba+SWA	RAMba
Total Params (M)	372	368	375	375	385
Hidden size, d	1024	1024	1024	1024	1024
Mamba Layers	-	48	48	48	48
Attention Layers	16	-	3×NSA	3×SWA	3×HSA
Other Layers	16×MLP	-	-	-	1×Chunk Selection, 1×Encoder
MLP hidden size	5504	-	-	-	1344
Query heads	32	-	16	-	16
KV heads	16	-	1	-	1
Vocab size	50280	50280	50280	50280	50280

Table 8: Hyper-parameters of 370M models.

C Training hyper-parameters

All 370M models used the **AdamW** optimizer with

- linear learning rate warmup with warmup ratio 0.02, cosine decay to $4e-5$.
- peak learning rate $2e-3$.
- total tokens $60B$, batch size $1M$ tokens.
- gradient clip value 1.0
- no dropout
- no linear bias terms
- weight decay $1e-3$
- AdamW hyperparameter $\beta = (.9, .95)$ (the GPT3 value)

All models are pre-trained on 16 PPUs, with each taking approximately 60 hours.

D Templates for passkey retrieval and Single-NIAH tasks

The passkey retrieval template is structured as follows: "(essays) The pass key is <PASS KEY>. (essays) What is the passkey? The passkey is". The single NIAH template follows this format: "(essays)... One of the special magic numbers for long-context is: <PASS KEY>. (essays)... What is the special magic number for long-context mentioned in the provided text? Answer:"

E RAMba 2.7B

We follow the hyperparameters of Mamba-2 2.7B, where the embedding dimension is 2560, and the total number of layers is 64. We use a two-layer Transformer-based encoder with a hidden size of 2560 and an intermediate dimension of 3392. HSA is inserted into Mamba starting from the 32nd layer, with one HSA layer inserted every 8 layers. In HSA, the group size g is 5, with a query head size h of 16 for each group. The total additional parameters amount to approximately 110M.

E.1 Post-training

Due to the length generalization issues inherent in the Mamba-2 2.7B, we post-train it to stabilize its perplexity on longer contexts. Specifically, we utilized BPTT for post-tuning the base model. We trained the model on sequences of 32K tokens with a batch size of 16 for 3K steps, totaling 1.5B tokens. This stage takes 5 hours on 32 PPUs.

Then we follow CEPE [69] by freezing most parameters of the Mamba backbone and tuning the parameters of the additional HSA modules for post-training. This process primarily involves the following two stages:

Warmup. We employ a warmup initialization method by simply training the model to copy the first half of the sentence. Specifically, we append an identical copy of each sentence to itself and train the model to locate the distant context and replicate it. At this stage, all parameters of Mamba-2 are frozen, with only the HSA-related parameters remaining tunable. We train the model with a 32K context length, batch size of 16, for 16K steps, with a peak learning rate of 2×10^{-5}, totaling 8B tokens. This stage takes around 24 hours on 32 PPUs.

Post-Training. We use LoRA [26] to fine-tune 5% of the parameters in the Mamba-2 module, while keeping all parameters in the HSA module fully trainable. The model is trained with a context length of 32K, a batch size of 16, for 32K steps, using a peak learning rate of 2×10^{-5}, totaling 16B tokens. This stage takes around 48 hours on 32 PPUs.

This model is used for the LongBench evaluation in Table 5.

E.2 RULER finetuning

Since the RULER tasks like NIAH and FWE do not have high requirements for intrinsic knowledge of LLMs, we opt to train RAMba from scratch and fine-tune it on RULER's synthetic dataset. This approach aims to evaluate whether RAMba trained from scratch can stably converge and demonstrate long-range retrieval capabilities. We conduct pre-training on 60B tokens, which amounts to one-tenth of the Mamba-2 2.7B model, followed by fine-tuning on 1B synthetic data, which takes around 200 hours on 32 PPUs. We also fine-tuned the Mamba-2 2.7B model on the same synthetic dataset for comparison.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in abstract and introduction are supported by experimental results in Section 4.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitation section is presented in page 16.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.

- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification:

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: This paper provides detailed experiment settings in Section 4.1, Appendix A,B, and D, which should be sufficient for reproducing the main experimental results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often

one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The sources of all data are explained in the paper, and the code will be open-sourced on GitHub after de-anonymization.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: This paper provides detailed experiment settings in Section 4.1, Appendix A,B, and D, which should be sufficient for reproducing the main experimental results.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Since each pre-trained model only has a single checkpoint, it is not feasible to evaluate error bars. However, in all experiments, we include multiple datasets or multiple sets of tasks and evaluate using various metrics. Therefore, the statistical significance of the experiments can be verified and supported.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: In Appendix B,D and Section 4.4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: Our work adheres to all the ethical guidelines outlined by NeurIPS.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All dataset details and original authorship are cited in Section 4.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The code will be published and well documented after de-anonymization.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our paper does not involve crowdsourcing nor research with human subjects, hence do not require IRB approval.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: Our core method doesn't involve the usage of LLMs.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.