Expansion Span: Combining Fading Memory and Retrieval in Hybrid State Space Models

Elvis Nunez ELVISNUN@AMAZON.COM
Luca Zancato ZANCATO@AMAZON.COM
Benjamin Bowman BOWMABEN@AMAZON.COM
Aditya Golatkar AGOLATKA@AMAZON.COM
Wei Xia WXIA@AMAZON.COM
Stefano Soatto SOATTOS@AMAZON.COM
AWS AI Labs

Editors: G. Pappas, P. Ravikumar, S. A. Seshia

Keywords: Hybrid State Space Models, SSMs, Attention, Sparse Attention, LoRA.

Abstract

The "state" of State Space Models (SSMs) represents their memory, which fades exponentially over an unbounded span. By contrast, Attention-based models have "eidetic" (i.e., verbatim, or photographic) memory over a finite span (context size). Hybrid architectures combine State Space layers with Attention, but still cannot recall the distant past and can access only the most recent tokens eidetically. Unlike current methods of combining SSM and Attention layers, we allow the state to be allocated based on relevancy rather than recency. In this way, for every new set of query tokens, our models can "eidetically" access tokens from beyond the Attention span of current Hybrid SSMs without requiring extra hardware resources. We introduce a method to expand the memory span of the hybrid state by "reserving" a fraction of the Attention context for tokens retrieved from arbitrarily distant in the past, thus expanding the eidetic memory span of the overall state. We call this reserved fraction of tokens the "expansion span," and the mechanism to retrieve and aggregate it "Span-Expanded Attention" (SE-Attn). To adapt Hybrid models to using SE-Attn, we propose a novel fine-tuning method that extends LoRA to Hybrid models (HyLoRA) and allows efficient adaptation on long spans of tokens. We show that SE-Attn enables us to efficiently adapt pre-trained Hybrid models on sequences of tokens up to 8 times longer than the ones used for pre-training. We show that HyLoRA with SE-Attn is cheaper and more performant than alternatives like LongLoRA when applied to Hybrid models on natural language benchmarks with long-range dependencies, such as PG-19, RULER, and other common natural language downstream tasks.

1. Introduction

State Space Models are able to process sequences with an unbounded number of tokens by maintaining a fixed-size state. However, this state is lossy and information about early tokens "fades" as more inputs are processed. In contrast, Transformer models have a state determined by the number of tokens in their input sequence and are able to access information from all past tokens in their context "eidetically." However, they do so at the cost of extra compute and memory.

Recent Hybrid models Zancato et al. (2024); Glorioso et al. (2024b); De et al. (2024) augment SSMs with Attention layers in an effort to counteract SSMs' "fading" memory. However, Attention layers only aggregate information by recency (i.e., they process the keys and values of the most recent tokens up to hardware limitations). Hence, any token that lies beyond the Attention's limited

span can only be approximately recalled through the SSMs' state. This limits the effectiveness of Hybrid SSMs when applied to long sequences of tokens, especially when compute and memory resources are limited. In fact, while recent pre-trained Hybrid models have demonstrated strong performance on common downstream and recall-intensive language tasks, they are trained following Transformers' pre-training practice: they are either trained on a fixed, relatively short window of tokens when compute is limited—which is often set to 2048 as in LLaMA1 Touvron et al. (2023a), or 4096 as in LLaMA2 Touvron et al. (2023b)—or are progressively fine-tuned on longer sequences after the pre-training stage Lieber et al. (2024); Glorioso et al. (2024b); Dubey et al. (2024).

In this work, we modify Hybrid SSMs' Attention layers to allow their state to be allocated by *relevancy* rather than *recency*, allowing our models to retrieve information that would otherwise fall outside their Attention span. To do so, we propose Span-Expanded Attention (SE-Attn), a selection mechanism that enables eidetic retrieval of tokens from the past. While SE-Attn can be applied to both Hybrid SSMs and Transformers, we focus on hybrid architectures where this retrieval capability naturally complements the fading memory of SSMs: while SSMs efficiently process long sequences with their state-based memory, SE-Attn selectively preserves important information that would otherwise fade in the SSM state.

Our method allows long context fine-tuning of any pre-trained Hybrid model, such as Mamba-2-Hybrid Dao and Gu (2024), Jamba Lieber et al. (2024), and Zamba Glorioso et al. (2024a), on sequences longer than the one used for pre-training without significant compute/memory overhead. While Transformer-based fine-tuning strategies for processing long sequences can also be applied to Hybrid models, they typically assume that information in the Attention layers is aggregated by recency and are thus more expensive as sequences get longer. For example, extending a LLaMA model from a 2k to 8k context size can require up to 32 A100 GPUs Chen et al. (2023). While there exist efficient LoRA-based methods Hu et al. (2022) that lower this cost, we show that Transformer-based methods tend to not work as well for Hybrid models as they do for Transformers. We show that this is mostly due to the fact that combining LoRA (on Attention layers) with SSM layers is not expressive enough to model long-range dependencies beyond the pre-training context size.

To overcome such limitations, we propose to modify the standard LoRA recipe to fit the Hybrid model structure better. Namely, we employ a fine-tuning strategy similar to LoRA+ Chen et al. (2024) on Attention layers, while allowing the SSMs' recurrent parameters to be adapted as well. In particular, building upon previous observations Zancato et al. (2024); Yang et al. (2024b), we also adapt the 1D convolutional layers; we empirically demonstrate that this adaptation yields the best results at a reduced cost.

We make the following contributions: (1) We propose SE-Attn, a novel selection mechanism that "reserves" a fraction of the context of Hybrid SSMs' Attention layers for tokens that are retrieved from the past based on their relevance to the current query. (2) We empirically show that Transformer-based methods for long-context fine-tuning are sub-optimal when applied to Hybrid SSMs. Therefore, we propose HyLoRA, an extension of LoRA+ Chen et al. (2024) for efficient and effective fine-tuning of Hybrid SSMs. (3) We show that combining SE-Attn and HyLoRA on Hybrid models reliably extends their context size up to $8\times$ their pre-training size. Further, we show improved performance over existing adaptation methods on common natural language tasks in the LM Harness Evaluation suite Gao et al. (2024) and long context tasks in RULER Hsieh et al. (2024).

2. Background and Related Work

Attention on long contexts. Since the introduction of Self-Attention Vaswani et al. (2017), a significant amount of research has been made to reduce its quadratic cost in the sequence length at training time. To achieve this, state of the art methods usually approximate the Attention mechanism with sparse/linearized versions. For example, Reformer Kitaev et al. (2020) uses locality-sensitive-hashing to group tokens with similar embeddings, allowing the model to only attend to a subset of tokens rather than the entire sequence. Other works have proposed to endow Transformers models with "compressed" memory tokens that are updated dynamically and causally over sliding windows on entire sequence chunks. For example, Transformer-XL Dai (2019) and Infini-Attention Munkhdalai et al. (2024) segment an input sequence into chunks and process them sequentially while maintaining a complementary set of tokens whose purpose is to summarize the older ones. In contrast to these works, our SE-Attn retrieves relevant tokens "eidetically," i.e., we retrieve tokens rather than maintain a compressed representation of the past. Most similar to our method is Landmark Attention Mohtashami and Jaggi (2023), which inserts landmark tokens into the input at fixed block intervals and trains these tokens to act as summaries of their corresponding blocks via a grouped softmax attention; these summary tokens are then used to index and retrieve relevant input tokens when processing future segments. Our SE-Attn aims to integrate retrieval natively, without the need for external landmark tokens or complex Softmax procedures.

State Space Models. While a great effort has been made to improve the efficiency of Transformer models, a recent line of work has explored efficient alternative "linear" architectures. In particular, State Space Models (SSMs) Gu and Dao (2023); Gu et al. (2021, 2022); Yang et al. (2024a); Sun et al. (2023) have emerged as promising competitors to Transformer models due to their efficient scaling and strong empirical performance. Numerous variations of SSMs have been proposed, some closely resembling Linear Attention Sun et al. (2023) or Linear-Time Invariant dynamical systems Gu et al. (2021), while others introduce novel adaptive/gated state updates Yang et al. (2024a); Gu and Dao (2023); Dao and Gu (2024); Orvieto et al. (2023). Despite their differences, all follow the same basic working principle that is inspired by classical state space models Kalman (1960): they process the input sequence by maintaining a *fixed-size* state which acts as a compressed (lossy) representation of all the processed tokens. When implemented in hardware, the state must have finite precision and therefore "fades" as more samples are processed. To overcome this limitation, the most successful SSMs are typically hardware-aware implementations that efficiently utilize modern GPUs/TPUs. These implementations use highly parallelizable and scalable primitives, such as associative scans Gu and Dao (2023); De et al. (2024), chunking mechanisms Dao and Gu (2024); Yang et al. (2024a), and techniques that avoid the materialization of the entire state in slow high bandwidth memory Gu and Dao (2023).

Hybrid State Space Models. While extending the recurrent state in SSMs layers has led to performant models, they typically fall short on tasks that require recalling information from the distant past Waleffe et al. (2024); Jelassi et al. (2024). Hybrid State Space Models have been introduced to solve this limitation and are typically designed to complement the SSMs' "fading" state with Attention layers Dao and Gu (2024); De et al. (2024); Lieber et al. (2024); Glorioso et al. (2024b). While early architectures simply stack SSMs and Attention layers with different blending ratios Waleffe et al. (2024); Gu and Dao (2023); Dao and Gu (2024) or replace full Attention layers with Sliding Window Attention De et al. (2024), more performant and sophisticated hybrid architectures have been recently proposed Glorioso et al. (2024b); Zancato et al. (2024). In particular,

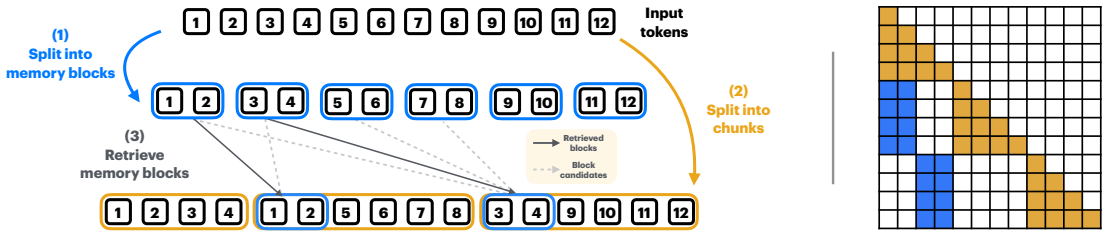


Figure 1: **Span-Expanded Attention (SE-Attn) overview.** SE-Attn is a Sparse Attention mechanism used to expand the memory span of Hybrid SSMs. *Left*: SE-Attn works by reserving a fraction of the Attention context for tokens retrieved arbitrarily far back in the past. We call this reserve the "expansion span," and we populate it with blocks of previous tokens ("memory blocks"). When new tokens arrive, a similarity-based search compares the queries with past memory blocks—represented as summary tokens—to retrieve relevant memory blocks. Then, these retrieved memory blocks are jointly processed with the queries via Attention. While the final Attention mechanism always processes a fixed number of tokens, it can have a longer span since it retrieves tokens from arbitrarily far back in the past. *Right*: Retrieving tokens from the past yields a sparse Attention pattern.

B'MOJO Zancato et al. (2024) complements SSMs' fading state with a form of "eidetic" memory whereby SSM layers are mixed with Sliding Window Attention and tokens in the window can attend to a set of tokens from the past that are deemed difficult to predict using an asynchronous causal selection mechanism. Similarly, in our work, we mix SSM layers with an Attention layer that operates over chunks of tokens, but we enable these chunks to dynamically retrieve tokens from the past that are most relevant to the chunk's tokens, rather than tokens selected a priori as in B'MOJO.

3. Span-Expanded Attention

Our goal is to enable pre-trained Hybrid SSMs to accurately process sequences longer than the ones seen during pre-training. For a sequence of L tokens with model dimension d_{model}, the computational complexity of standard Self-Attention is $O(d_{\text{model}}L^2)$. For very large L, computing Self-Attention can be prohibitively expensive. Training models with a large L is particularly challenging, as training has the additional cost of computing gradients, further limiting memory resources. To address this, we propose Span-Expanded Attention (SE-Attn), a drop-in replacement for standard Self-Attention in Hybrid SSMs. To train SE-Attn, we propose HyLoRA, a variant of LoRA+ Chen et al. (2024) specifically tailored to Hybrid models. A schematic of our SE-Attn is provided in Figure 1.

3.1. Attention

At the heart of modern LLMs is the Attention mechanism, whose role is to construct a contextualized representation of its input. The input to Attention, $x \in \mathbb{R}^{L\times d}$, is a tensor of L tokens, each with dimension d. Attention is parameterized by $W_Q, W_K, W_V \in \mathbb{R}^{d\times d_{\text{model}}}$ and $W_o \in \mathbb{R}^{d_{\text{model}}\times d}$, which are used to construct linear projections of the input: $Q = xW_Q$, $K = xW_K$, $V = xW_V \in \mathbb{R}^{L\times d_{\text{model}}}$. After adding positional information to the keys and queries, the output of the Attention layer is[1] $\text{Attention}(Q,K,V) = \left[\text{softmax}\left(\frac{QK^T}{\sqrt{d_{\text{model}}}}\right)V\right]W_o \in \mathbb{R}^{L\times d}$. The realization of the attention score

1. For simplicity, we consider Single-Head Attention in this exposition.

matrix, $QK^T \in \mathbb{R}^{L\times L}$, grows quadratically in the sequence length, L. Moreover, the output of Attention is typically followed by a feed-forward layer (FFN), which expands and contracts the dimension of the input. While FFNs are generally regarded as being the primary computational bottlenecks of Attention-based models for short contexts, when the sequence length exceeds the expanded dimension, Attention becomes the primary bottleneck. Since LLMs consist of many layers of Attention, this computation is particularly expensive during training, where activations are cached. To make the Attention mechanism computationally amenable to longer contexts during training, we draw inspiration from RAG and Sparse Attention methods.

3.2. Amnesic Attention

The crux of our method is in the chunking of the L tokens in the input sequence x, into chunks of size M. Namely, we begin by computing projections $Q = xW_Q$, $K = xW_K$, $V = xW_V$ and adding positional information as in standard Attention. Next, each of the $Q, K, V \in \mathbb{R}^{L\times d_{\text{model}}}$ projections are split into $T = \frac{L}{M}$ chunks of M tokens in each along the sequence dimension, yielding $Q_i, K_i, V_i \in \mathbb{R}^{M\times d_{\text{model}}}$ for $i = 1, \ldots, T$. A naive way to reduce the quadratic cost of Attention is to apply it independently on each of these chunks, $A_i = \text{Attention}(Q_i, K_i, V_i)$ and then concatenate and project them to get the final output, $\text{Concatenate}(A_1, A_2, \ldots, A_T)$. However, since the chunks are processed independently, there is no information exchanged between them. Hence, this naive—though efficient—Attention mechanism, which we refer to as "SE-Attn-NoMem", cannot model time dependencies on contexts larger than the chunk size.

3.3. Eidetic Retrieval Attention

In this section, we improve upon SE-Attn-NoMem by allowing different chunks to exchange information while minimizing compute; we call this Attention mechanism SE-Attn. To this end, we augment the processing of each chunk with a mechanism to retrieve tokens from previous chunks. In particular, we allow chunk i to retrieve tokens from chunks $1, 2, \ldots, i-1$ and, when the most relevant chunks are selected, append their tokens to its context (its "expansion span"). SE-Attn takes as input a sequence, $x \in \mathbb{R}^{L\times d}$ and computes projections $Q = xW_Q$, $K = xW_K$, $V = xW_V$ followed by the addition of RoPE Su et al. (2024) embeddings as in standard Attention. As described in Section 3.2, Q, K, V are split into T chunks with M tokens in each, yielding tuples (Q_i, K_i, V_i), $i = 1, \ldots, T$. Additionally, Q, K, V are split into a second set of U chunks with S tokens in each, yielding tuples $(Q_j^{\text{Mem}}, K_j^{\text{Mem}}, V_j^{\text{Mem}})$ where $Q_j^{\text{Mem}}, K_j^{\text{Mem}}, V_j^{\text{Mem}} \in \mathbb{R}^{S\times d_{\text{model}}}$ for $j = 1, \ldots, U$. We refer to these tuples as "memory blocks"; in SE-Attn, the query from each chunk, Q_i, attends not only to K_i, but also to a set of retrieved memory blocks which populate SE-Attn's expansion span. In particular, each Q_i retrieves k (top-k) key/value memory blocks from the past[2]: $\left(K^{\text{Mem}}_{\phi_i(U)_1}, V^{\text{Mem}}_{\phi_i(U)_1}\right), \ldots, \left(K^{\text{Mem}}_{\phi_i(U)_k}, V^{\text{Mem}}_{\phi_i(U)_k}\right)$ where $\phi_i(U)_j$ denotes the index of the j-th memory block selected by the i-th chunk. Retrieved blocks are appended to the chunks' keys and values, and SE-Attn computes the Attention output for the ith chunk as in Equation (3).

$$\tilde{K} = \text{Concatenate}(K^{\text{Mem}}_{\phi_i(U)_1}, \ldots, K^{\text{Mem}}_{\phi_i(U)_k}, K_i) \tag{1}$$

$$\tilde{V} = \text{Concatenate}(V^{\text{Mem}}_{\phi_i(U)_1}, \ldots, V^{\text{Mem}}_{\phi_i(U)_k}, V_i) \tag{2}$$

$$A_i^{\text{SE-Attn}} = \text{Attention}(Q_i, \tilde{K}_i, \tilde{V}_i) \tag{3}$$

2. By "the past," we mean tokens in the sequence that came before tokens in the chunk.

Afterwards, each chunk's Attention outputs are concatenated and projected:

$$o^{\text{SE-Attn}} = \text{Concatenate}(A_1^{\text{SE-Attn}}, A_2^{\text{SE-Attn}}, \ldots, A_T^{\text{SE-Attn}}).$$

Note that Equation (3) is a form of Cross-Attention since we are not concatenating memory query tokens to the chunk's query. This is to preserve causality, as the retrieved tokens cannot attend to the chunk's tokens.

Memory Retrieval. Each chunk x_i must judiciously select which memory blocks to retrieve from the past. To do this efficiently, we associate a 1-dimensional tensor, $c_j \in \mathbb{R}^{d_{\text{model}}}$, to each of the $j = 1, \ldots, U$ memory blocks which act as a compressed representation of each memory block. To determine which memory blocks Q_i should attend to, we compute a "relevancy score" between Q_i and each c_j, which measures how relevant memory block j is to chunk i. This relevancy score is implemented with a Cross-Attention score between chunks and compressed memory block representations. Recalling that $Q_i \in \mathbb{R}^{M \times d_{\text{model}}}$, we compute relevancy score $R_{ij} \in \mathbb{R}$ between chunk $i \in \{1, 2, \ldots, T\}$ and memory block $j \in \{1, 2, \ldots, U\}$ as follows: $R_{ij} = \sum_{t=1}^{M} (Q_i c_j)_t$. $R_i \in \mathbb{R}^U$ represents the relevancy score between chunk i and all memory blocks. However, since chunk i should only retrieve memory blocks that came before it temporally, we add a mask to R_i and then apply Softmax to obtain the final scores, $\tilde{R}_i$, between chunk i and all memory blocks as follows:

$$\tilde{R}_i = \text{softmax}\left(\frac{1}{\sqrt{d_{\text{model}}}}(R_i + \mathcal{M}_i)\right) \in \mathbb{R}^U \tag{4}$$

where $\mathcal{M}_i$ is a mask of 0 and $-\infty$ constructed to set the scores of future memory blocks (relative to chunk i) to $-\infty$. Once all relevancy scores are computed, chunk i simply retrieves the top k memory blocks with the highest $\tilde{R}_{ij}$ scores and concatenates them with the keys and values as in Equations (1) and (2) before computing Attention as in Equation (3).

Compressed Memory Blocks. Next, we discuss how we construct the compressed memory block representations, c_j. Recently, Landmark Attention Mohtashami and Jaggi (2023) considered using "landmark" tokens to obtain compressed representations of memory blocks. We consider using landmark tokens to construct c_j in Appendix F. In SE-Attn, we consider a simpler approach which we found to work well. For each memory block, $(Q_j^{\text{Mem}}, K_j^{\text{Mem}}, V_j^{\text{Mem}})$, we perform standard non-causal Attention, $A_j^{\text{Mem}} = \text{softmax}\left(\frac{Q_j^{\text{Mem}}(K_j^{\text{Mem}})^T}{\sqrt{d_{\text{model}}}}\right) V_j^{\text{Mem}} \in \mathbb{R}^{S \times d_{\text{model}}}$; we consider non-causal Attention as we are interested in a global representation where all tokens within the memory block can attend to each other. We compute the mean of these weighted memory tokens as the compressed representation: $c_j = \frac{1}{S}\sum_{t=1}^{S}(A_j^{\text{Mem}})_t \in \mathbb{R}^{d_{\text{model}}}$, where $(A_j^{\text{Mem}})_t$ denotes the t-th row of A_j^{Mem}.

3.4. Training SE-Attn with LoRA

In this paper, we fine-tune models pre-trained with standard Attention; however, we fine-tune them with SE-Attn, which modifies the pre-trained Attention mechanism by introducing a retrieval mechanism. SE-Attn repurposes the model's Attention parameters (W_Q, W_K, W_V, W_o) to perform retrieval. In order to efficiently train the model to learn to use SE-Attn, we use a variant of LoRA Hu et al. (2022). Recently, Chen et al. (2024) introduced LoRA+ designed to fine-tune Transformer models on long contexts. LoRA fine-tunes Attention parameters with low rank adapter matrices. LoRA+ differs from LoRA by also training embedding and normalization layers.

Recently, Galim et al. (2024) found that pure SSMs can be fine-tuned by training the SSM projection layers with LoRA. Since we fine-tune on long contexts, we prioritize efficient training, and consequently apply LoRA only to the Attention layers of our hybrid models. However, it is common for SSM layers to include a 1D convolution layer after their initial projection in order to perform sequence mixing Zancato et al. (2024); Glorioso et al. (2024b); Lieber et al. (2024); Dao and Gu (2024); De et al. (2024). The 1D convolution parameters constitute a very small portion of the model parameters ($\sim$0.7% for Mamba-2-Hybrid 2.7B), but as discussed further in Section 4.3, we found that training these 1D convolution layers in conjunction with LoRA+ improved our models' performance on long-context tasks; we refer to this augmented LoRA+ variation as HyLoRA.

4. Experiments

4.1. Experimental Setup

Models. We enhance the recall capabilities of pre-trained hybrid SSM models by fine-tuning them with different Attention layers on spans of tokens longer than the ones used for pre-training. We consider Mamba-2-Hybrid 2.7B Waleffe et al. (2024) as our representative SSM hybrid model and provide results for Zamba2-Hybrid Glorioso et al. (2024a) in Appendix D. We also explore expanding the span of Transformer models in Appendix C. These models, obtained from Hugging Face, were pre-trained with a context size of 2048. In our experiments, "Non-fine-tuned" refers to the pre-trained model with no fine-tuning. Similar to Chen et al. (2024), we utilize SE-Attn for efficient fine-tuning and revert to using Full-Attn during evaluation; this is discussed further in Appendix A.
Datasets. We fine-tune models using a common language dataset and provide additional results when fine-tuning on PG-19 Rae et al. (2019a) and a mixture of language and code dataset in Appendix E.
Baselines. We compare SE-Attn to three Attention variants: standard full Attention ("Full-Attn") Vaswani et al. (2017), Sliding Window Attention ("SW-Attn") Beltagy et al. (2020), and Shifted Sparse Attention (S^2-Attn) Chen et al. (2024). Full-Attn serves as the paragon, as the other methods aim to approximate it. All Attention implementations use FlashAttention-2 Dao (2024).
Training Procedure. We adopt the training recipe used in Chen et al. (2024) to fine-tune our models, with the exception of using HyLoRA instead of LoRA+. See Appendix G for more details.
Evaluation Metrics. We assess the performance of our models across a range of common benchmarks. To assess the predictive capabilities of our fine-tuned models on long sequence lengths, we measure perplexity (PPL) on the PG-19 validation set; however, as discussed further in Appendix B, we do not find PPL to be a faithful measure of a model's long context abilities. To assess performance on more real-world language tasks in the short and long-context settings, we evaluate our models on various LM Evaluation Harness tasks Gao et al. (2024). To measure our models' performance on long-context tasks, we evaluate on the RULER Hsieh et al. (2024) benchmark. We consider eleven RULER tasks, encompassing needle-in-a-haystack, variable-hopping, and aggregation tasks; we aggregate these metrics into five groups as explained in Appendix I.4. We provide additional long-context benchmarks in Appendix E.3.

4.2. Results

All of our Mamba-2-Hybrid models are fine-tuned with a context size of 8192. For SE-Attn, we use a block size of 32, and a top-k of 8. For SW-Attn, we use a window size of 4096. S^2-Attn uses the parameters from Chen et al. (2024). When fine-tuning Mamba-2-Hybrid using SE-Attn, we found that applying the same chunk sizes at each layer leads to suboptimal downstream performance. Therefore, we segment each sample into chunks of variable sizes (picked randomly from $\{2048, 4096\}$). We

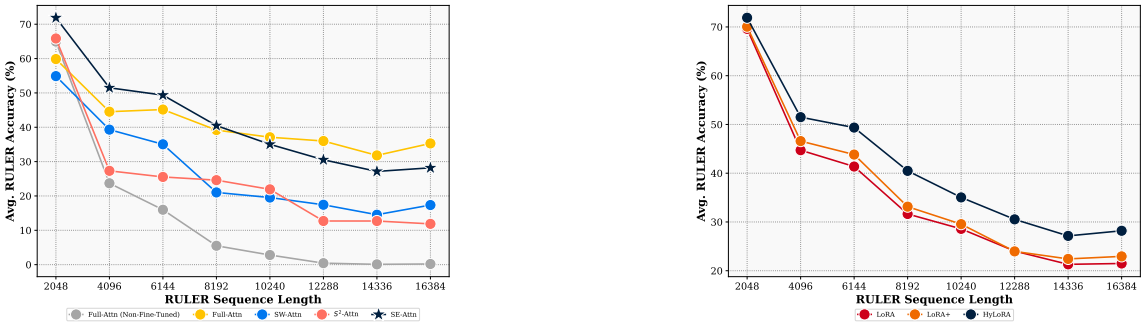
Attention	Eval Context Size (PG-19 PPL ↓)				Short Context Tasks (↑)							Long Context Tasks (↑)			
	2048	8192	16384	32768	ARC-E	ARC-C	Hella.	LAMB.	PIQA	WG	Avg.	SWDE	SQA	SNQA	Avg.
Non-fine-tuned	10.72	14.99	19.35	26.37	69.91	37.97	67.62	69.84	76.06	65.04	64.41	85.60	15.18	3.65	34.81
Full-Attn	10.99	10.28	10.39	11.14	69.53	38.48	67.30	68.93	75.08	64.40	63.95	85.24	26.99	19.75	43.99
SW-Attn	10.98	10.80	11.82	13.45	69.82	38.23	67.35	69.18	75.30	63.85	63.95	84.61	24.85	15.41	41.63
S^2-Attn	10.87	12.89	14.67	16.37	70.12	38.05	67.39	69.84	75.95	64.56	64.32	86.41	17.44	8.53	37.46
SE-Attn	10.99	10.45	11.14	12.64	70.20	38.65	67.15	69.11	75.57	63.93	64.10	85.96	26.70	18.04	43.57

Table 1: **Fine-tuning Mamba-2-Hybrid with SE-Attn outperforms fine-tuning with S^2-Attn and SW-Attn on long-context natural language tasks.** We evaluate PG-19 validation perplexity (PPL) and observe that fine-tuning with SE-Attn preserves performance at longer contexts better than S^2-Attn and SW-Attn. On long context tasks from the LM Harness suite, SE-Attn outperforms S^2-Attn and SW-Attn. See Appendix J for task abbreviation definitions.

found this prevents the model from overfitting to a fixed chunk size, which is a hyperparameter chosen a priori, independent of the actual data content. An ablation on SE-Attn with different chunk sizes is provided in Section 4.3.

Perplexity. We provide PG-19 PPL results in Table 1. All fine-tuned models outperform the non-fine-tuned model. SE-Attn yields the closest performance to the paragon model fine-tuned with Full-Attn, outperforming S^2-Attn and SW-Attn across all context sizes at or above the fine-tuning size.

LM Harness. Next, we evaluate Mamba-2-Hybrid models across short and long-context tasks in the LM Evaluation Harness suite. Our results are provided in Table 1, where we observe that all models perform similarly on short-context tasks—including the non-fine-tuned model—suggesting there is no performance regression when fine-tuning on larger contexts. Furthermore, we observe that fine-tuning with SE-Attn gives the closest performance to fine-tuning with Full-Attn.

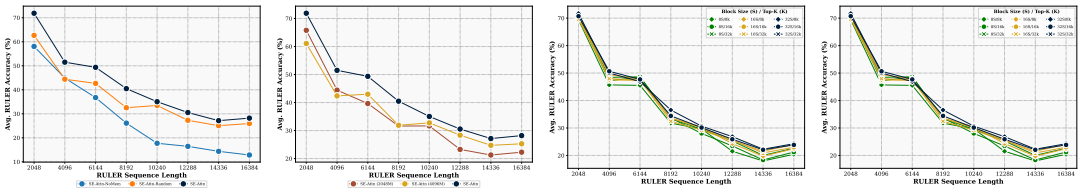


(*a*) Mamba-2-Hybrid Attention Variants (*b*) Mamba-2-Hybrid LoRA Variants

Figure 2: **Fine-tuning with SE-Attn outperforms SW-Attn and S^2-Attn on the RULER benchmark. HyLoRA outperforms LoRA and LoRA+ on Hybrid models.** (a): We fine-tune Mamba-2-Hybrid with a context size of 8192 and evaluate on eleven RULER tasks, as explained in Appendix I.4. Fine-tuning with SE-Attn consistently outperforms SW-Attn and S^2-Attn even when evaluating on context sizes beyond the fine-tuning size. (b): We fine-tune Mamba-2-Hybrid with SE-Attn using LoRA, LoRA+, and HyLoRA. LoRA and LoRA+ perform sub-optimally. Our HyLoRA additionally trains the 1D convolution layers and yields strong performance.

RULER. Next, we assess the performance of our models on long-context tasks using the RULER benchmark. Figure 2 shows the average accuracy across eleven RULER tasks. We observe that

fine-tuning with SE-Attn performs similar to fine-tuning with Full-Attn, and outperforms S^2-Attn and SW-Attn. We provide more detailed RULER results in Figure 8, where we see a substantial improvement on the variable tracking (VT) task, which may be attributed to SE-Attn's retrieval during fine-tuning.



(*a*) (*b*) (*c*) (*d*)

Figure 3: **SE-Attn ablations on Mamba-2-Hybrid.** (a): Attention-based memory retrieval (SE-Attn) improves upon no retrieval and random retrieval. (b): Using SE-Attn with a chunk size chosen randomly from $\{2048, 4096\}$ acts as a regularizer and outperforms SE-Attn with fixed chunk sizes of 2048 and 4096. (c): SE-Attn with larger memory blocks (i.e., more tokens per block) with a smaller top-k tends to do better than smaller blocks with a larger top-k. (d): An expansion span consisting of 256 total tokens (8 memory blocks with 32 tokens in each) gives the strongest performance.

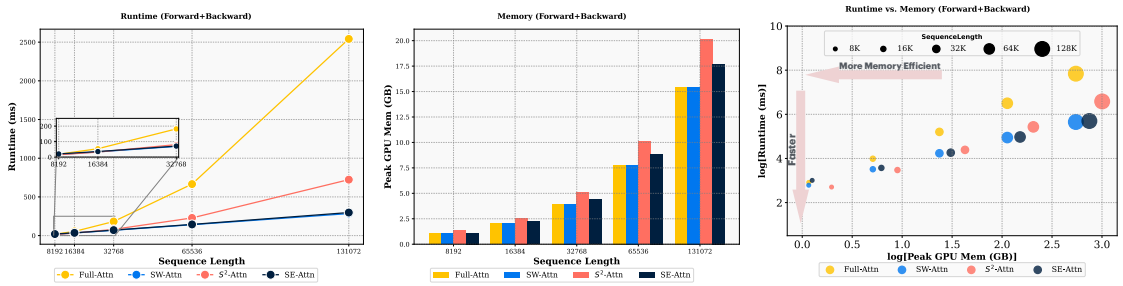
4.3. Ablations

In this section, we ablate over some of the design choices of SE-Attn on Mamba-2-Hybrid. For this analysis, we consider the RULER benchmark, as it is a strong indicator of performance on long-context tasks. We use the average of the eleven RULER tasks defined in Appendix I.4.

Does retrieval during training help? In Section 3.2, we introduced SE-Attn-NoMem, a variant of SE-Attn where we do not do any retrieval and process chunks independently. Naturally, we do not expect this to do well due to the lack of shared information across chunks. We confirm this in Figure 3(a), where we observe that SE-Attn-NoMem achieves a much lower performance than SE-Attn with retrieval. Furthermore, we also fine-tune with SE-Attn-Random, a variant of SE-Attn that populates its expansion span by retrieving random memory blocks for each chunk. This improves upon SE-Attn-NoMem, indicating that retrieving some information from the past improves performance; however, it does not do as well as SE-Attn, which retrieves blocks based on relevancy.

Chunk size. We found that using a random chunk size during each forward pass improved upon having a fixed chunk size. In Figure 3(b), we compare the performance of SE-Attn, which chooses a random chunk size in $\{2048, 4096\}$ during each layer's forward call, to SE-Attn with fixed chunk sizes of 2048 and 4096. Random chunk sizes outperform fixed ones, suggesting a regularizing effect that makes the model more robust to different context lengths.

Block size and top-k. For a fixed expansion span size, we might expect that retrieving memory blocks at a finer granularity should perform better than retrieving a smaller number of large blocks; this is because retrieving a larger number of small blocks is as flexible as retrieving a small number of large blocks. However, as illustrated in Figure 3(c), we found that fine-tuning SE-Attn with larger block sizes and a smaller top-k gave the best performance. This is possibly due to the increased complexity of the retrieval task, which gets more difficult as the number of possible blocks to retrieve increases. As we are fine-tuning with LoRA, we hypothesize that the model's capacity may not be



(*a*) (*b*) (*c*)

Figure 4: **SE-Attn offers a greater runtime-memory trade-off than other Attention variants.** We profile various attention layers during a single training step. SW-Attn uses a window size of 4096 and SE-Attn alternates between a chunk size of 2048 and 4096. (a): Runtime of a single training step. (b): Peak GPU memory used during the training step. (c): Runtime vs. memory used. Points on the lower left of the plot exhibit a stronger runtime-memory trade-off.

sufficient to learn to retrieve effectively. As illustrated in Figure 3(d), we found that an expansion span populated with 256 retrieved tokens (32 memory blocks with 8 tokens in each) works best.

HyLoRA for Hybrid Models. We fine-tune our hybrid models using HyLoRA, which builds upon LoRA+ Chen et al. (2024) by also training 1D convolution layers. In Figure 2(b) we show that, when evaluated on RULER tasks, fine-tuning SE-Attn models with HyLoRA gives the strongest performance. In Appendix B, we study the effect of the LoRA rank and apply HyLoRA to Full-Attn.

4.4. Empirical Runtime Analysis

While hardware-efficient Attention implementations, such as FlashAttention Dao (2024), enable linear memory scaling, computational complexity remains quadratic in sequence length. In Figure 4(a), we profile different Attention layers on various context sizes for one training step and see that SE-Attn is much faster than S^2-Attn and Full-Attn. Moreover, the runtime of SE-Attn is similar to that of SW-Attn, despite SE-Attn's much longer Attention span. We also note that the runtime increase of Full-Attn/FlashAttention compared to SE-Attn is 4-5$\times$ for long sequences, while the memory peak increase is 17%, as shown in Figure 4(b). SE-Attn has a minor memory overhead because it requires storing partial block summary statistics in order to retrieve tokens from the past. However, this overhead is outweighed by its faster runtime. As depicted in Figure 4(c), our method has a strong runtime-memory trade-off—similar to that of SW-Attn, but with the added benefit of improved performance on language benchmarks. See Appendix H for a theoretical runtime analysis.

5. Conclusion

We close with the limitations of our method. Although we have conducted experiments at a relatively large model scale (2.7B) and long contexts (up to 32k), testing our method for efficiently adapting larger models on longer contexts is paramount. Furthermore, while we utilize datasets that require modeling of long-range dependencies, we found that perplexity-based tasks do not faithfully measure models' capabilities to handle long contexts, and instead tasks like RULER provide better signals of long-context capabilities. However, RULER is mostly a synthetic dataset and does not cover more nuanced tasks that require reasoning over long documents. Validating our method on more complex long-range benchmarks is a promising area for future work.

References

Simran Arora, Aman Timalsina, Aaryan Singhal, Benjamin Spector, Sabri Eyuboglu, Xinyi Zhao, Ashish Rao, Atri Rudra, and Christopher Ré. Just read twice: closing the recall gap for recurrent language models. *arXiv preprint arXiv:2407.05483*, 2024.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. 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 3119–3137, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172/.

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

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*, 2023.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. LongloRA: Efficient fine-tuning of long-context large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=6PmJoRfdaK.

Zihang Dai. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=mZn2Xyh9Ec.

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.

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

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Kevin Galim, Wonjun Kang, Yuchen Zeng, Hyung Il Koo, and Kangwook Lee. Parameter-efficient fine-tuning of state space models. *arXiv preprint arXiv:2410.09016*, 2024.

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

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

Paolo Glorioso, Quentin Anthony, Yury Tokpanov, James Whittington, Jonathan Pilault, Adam Ibrahim, and Beren Millidge. Zamba: A compact 7b ssm hybrid model. *arXiv preprint arXiv:2405.16712*, 2024b.

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

Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. Combining recurrent, convolutional, and continuous-time models with linear state space layers. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 572–585. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/05546b0e38ab9175cd905eebcc6ebb76-Paper.pdf.

Albert Gu, Karan Goel, and Christopher Re. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=uYLFoz1vlAC.

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

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 *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Samy Jelassi, David Brandfonbrener, Sham M. Kakade, and Eran Malach. Repeat after me: transformers are better than state space models at copying. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

Gregory Kamradt. Needle in a haystack - pressure testing llms., 2023. URL https://github.com/gkamradt/LLMTest_NeedleInAHaystack/tree/main.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *The International Conference on Learning Representations (ICLR)*, 2020.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, et al. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*, 2024.

Amirkeivan Mohtashami and Martin Jaggi. Random-access infinite context length for transformers. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

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

Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. In *International Conference on Machine Learning*, pages 26670–26698. PMLR, 2023.

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, Chloe Hillier, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. *arXiv preprint*, 2019a. URL https://arxiv.org/abs/1911.05507.

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*, 2019b.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506, 2020.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

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*, 2023b.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Roger Waleffe, Wonmin Byeon, Duncan Riach, Brandon Norick, Vijay Korthikanti, Tri Dao, Albert Gu, Ali Hatamizadeh, Sudhakar Singh, Deepak Narayanan, et al. An empirical study of mamba-based language models. *arXiv preprint arXiv:2406.07887*, 2024.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024a.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 115491–115522. Curran Associates, Inc., 2024b. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/d13a3eae72366e61dfdc7eea82eeb685-Paper-Conference.pdf.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=r8H7xhYPwz.

Luca Zancato, Arjun Seshadri, Yonatan Dukler, Aditya Golatkar, Yantao Shen, Benjamin Bowman, Matthew Trager, Alessandro Achille, and Stefano Soatto. B'mojo: Hybrid state space realizations of foundation models with eidetic and fading memory. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 130433–130462. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/eb6281d3f2cfe2eb8b1307227d3677b3-Paper-Conference.pdf.

Vladislav Zavadskyy. Lots of code, 2017. URL https://www.kaggle.com/datasets/zavadskyy/lots-of-code. Accessed: 2024-10-27.

Attention	Evaluation Context Size (PPL ↓)			
	8192	16384	32768	65536
Non-fine-tuned	14.99	19.35	26.37	34.51
Full-Attn	10.28	10.39	11.14	12.38
SW-Attn	10.33	10.22	10.16	10.16
S^2-Attn	10.73	10.76	11.85	13.72
SE-Attn	10.47	10.70	10.91	10.96

Table 2: **Mamba-2-Hybrid fine-tuned with SE-Attn or SW-Attn preserves perplexity up to $32\times$ the pre-training context size.** We fine-tune a Mamba-2-Hyrbid model (pre-trained on a context size of 2048) on a context size of 8192 using various Attention variants. We test on longer context sizes using the same Attention used during adaptation.

Appendix A. Evaluating with Full Attention

Our models are efficiently fine-tuned using SE-Attn and use Full-Attn during evaluation, as in Chen et al. (2024). Due to KV-caching, the complexity of Full-Attn scales linearly during evaluation, so an efficient Attention layer in this setting is not as crucial as in training. Nevertheless, for the sake of completeness, we begin by evaluating our models with the same Attention layer used during fine-tuning (i.e., we fine-tune and deploy our Hybrid model with SE-Attn). In Table 2, we experiment with models that use Full-Attn, SW-Attn, S^2-Attn, and SE-Attn; then, we evaluate perplexity on the PG-19 dataset. All models were fine-tuned with a context size of 8192. We observe that SW-Attn is best at preserving the perplexity on context sizes up to $32\times$ larger than the one used for pre-training. SE-Attn is also able to maintain a lower perplexity across longer context sizes, while S^2-Attn deteriorates much more quickly. As discussed further in the next section, perplexity may not be the most faithful metric for assessing models' long-context capabilities.

Appendix B. LoRA for Hybrid Models and the Pitfalls of Perplexity

In this section, we conduct a more thorough ablation study on the effect of training Mamba-2-Hybrid with different variants of LoRA, and provide further evidence that perplexity is not a reliable indictor of long context performance.

Fine-tuning with Full-Attn. We first expand upon Figure 2(b), where we showed that fine-tuning Mamba-2-Hybrid with SE-Attn using HyLoRA produced a stronger model than training with LoRA and LoRA+. In Figure 5, we provide a similar plot, but using Full-Attn during fine-tuning rather than SE-Attn. We observe a similar trend, where HyLoRA produces a stronger model than LoRA and LoRA+. In the remaining analyses in this section, we provide results for fine-tuning with SE-Attn, as we observed similar trends when fine-tuning with Full-Attn.

Effect of LoRA rank. We next explore how the rank used for LoRA fine-tuning affects downstream performance on the RULER task. Given its stronger performance, we focus on our HyLoRA. In our previous Mamba-2-Hybrid experiments, we used a LoRA rank (r) of 32 and an α of 64 (for our Transformer experiments, we used $r = 8$ and $\alpha = 16$ as in Chen et al. (2024)). In Figure 6, we plot the average RULER results for Mamba-2-Hybrid models fine-tuned with HyLoRA using different ranks (we scale α to maintain the ratio $\alpha/r = 2$). Here, we observe that training with a larger rank improves downstream performance, and we start to see some saturation around $r = 64$.

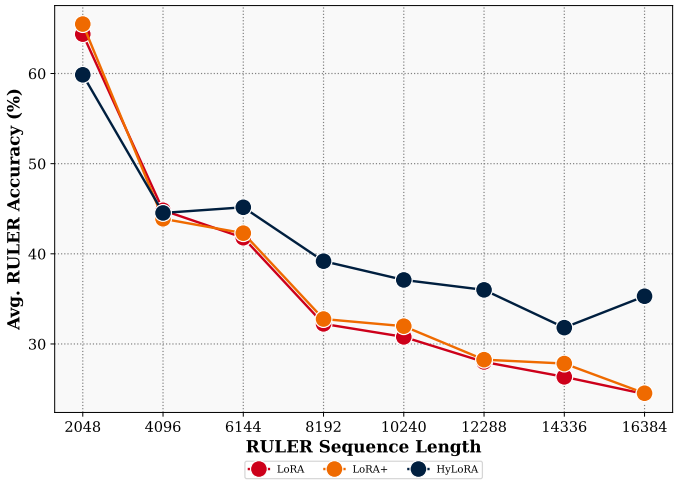


Figure 5: **HyLoRA outperforms LoRA and LoRA+ on Hybrid models.** We fine-tune Mamba-2-Hybrid with Full-Attn using LoRA, LoRA+, and HyLoRA. We find that LoRA and LoRA+ perform sub-optimally compared to HyLoRA which also trains 1D convolution layers.

In Table 3, we provide PG-19 perplexity results for Mamba-2-Hybrid trained with different LoRA variants. All models are fine-tuned with a context size of 8192 and evaluated with multiple context sizes. For a given context size, we do not see a substantial difference in the perplexity. This is similar to the observation in Chen et al. (2024). However, as discussed above, the rank can have a significant effect on downstream long-context tasks that require strong recall capabilities, as in RULER. Hence, while perplexity results may be promising, they are not necessarily indicative of performance on more complex long-context tasks.

Based on the analyses in this section, we conclude the following:

- Fine-tuning the 1D convolution layers, as we do in our HyLoRA, significantly improves performance on downstream long-context tasks that require retrieval, such as RULER.
- Fine-tuning with larger LoRA ranks improves performance up to a certain point—64 for our experiments.
- Perplexity may not be the most faithful metric for assessing performance on long-context downstream tasks. Instead, researchers should consider evaluating on more complex tasks, such as those in the RULER benchmark.

Appendix C. Applying SE-Attention to Transformers

In this section, we show that SE-Attn can also be applied to Transformer models and use Llama 1 7B Touvron et al. (2023a) as our representative Transformer model. When fine-tuning Llama 7B models, we found that using a fixed chunk size of $M = 4096$ gave better downstream performance. All of

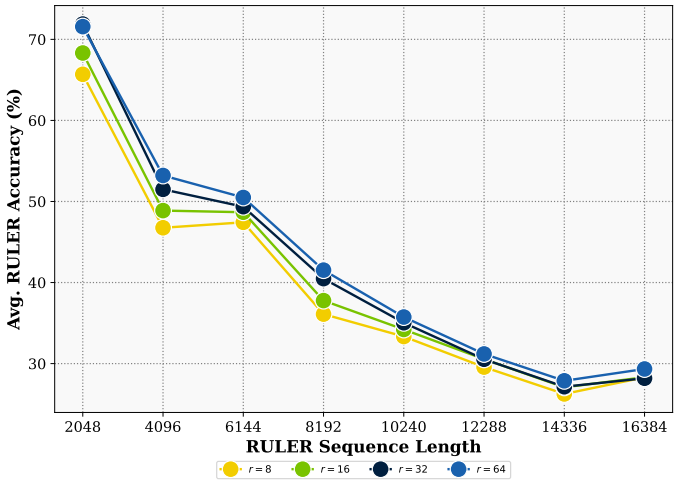


Figure 6: **Fine-tuning with a larger LoRA rank using HyLoRA improves performance on the RULER benchmark.** We fine-tune Mamba-2-Hybrid using HyLoRA with different LoRA ranks (we maintain a LoRA rank to alpha ratio of 2). We observe that fine-tuning with a larger rank produces stronger downstream results on RULER, with some saturation when using a rank of 64.

Sequence Length	LoRA Method (Rank=32)			LoRA Rank (Method=HyLoRA)			
	LoRA	LoRA+	HyLoRA	8	16	32	64
2048	10.77	10.98	10.99	11.00	11.00	10.99	10.99
8192	10.29	10.49	10.45	10.45	10.46	10.45	10.44
16384	10.91	11.18	11.14	11.03	11.10	11.14	11.14
32768	12.34	12.66	12.64	12.37	12.53	12.64	12.64
65536	13.99	14.36	14.26	13.84	14.08	14.26	14.28

Table 3: **Mamba-2-Hybrid LoRA Ablations.** We fine-tune Mamba-2-Hybrid with a context size of 8192 with SE-Attn using different LoRA variants. We consider LoRA, LoRA+, and HyLoRA (ours) and evaluate perplexity on the PG-19 dataset. We observe that all LoRA variants yield similar perplexity results. However, as depicted in Figure 2 and Figure 6, different LoRA variants yield substantially different performances on more complex tasks.

our Llama models are fine-tuned with a context size of 16384. For SE-Attn, we use a block size of 32, and a top-k of 8. For SW-Attn, we use a window size of 4096, and S^2-Attn uses the default parameters in Chen et al. (2024). Due to computational constraints, we do not consider fine-tuning with Full-Attn. All Attention variants use the same RoPE Su et al. (2024) scaling as in Chen et al. (2024).

Perplexity. Fine-tuning Llama with SE-Attn leads to better generalization on context sizes smaller and larger than the one used for fine-tuning. In Table 4, we observe that fine-tuning with SE-Attn preserves the performance of the non-fine-tuned model at smaller context sizes, and offers greater generalization to larger context sizes, as measured on PG-19 validation perplexity.

Attention	Eval Context Size (PG-19 PPL ↓)				Short Context Tasks (↑)							Long Context Tasks (↑)			
	2048	8192	16384	32768	ARC-E	ARC-C	Hella.	LAMB.	PIQA	WG	Avg.	SWDE	SQA	SNQA	Avg.
Non-fine-tuned	8.63	17.62	105.78	244.32	73.19	41.38	66.64	54.38	76.28	62.51	62.40	38.52	20.75	12.36	23.88
SW-Attn	8.80	8.17	8.35	10.32	74.20	43.09	75.00	71.96	77.42	67.48	68.19	82.81	24.41	21.06	42.76
S^2-Attn	9.32	8.64	8.58	10.42	74.20	42.58	73.84	69.94	77.80	67.72	67.68	80.56	23.36	19.65	41.19
SE-Attn	8.76	8.13	8.01	9.28	75.04	44.37	75.02	71.03	78.02	67.17	68.44	84.79	24.59	21.36	43.58

Table 4: **Fine-tuning Llama1 with SE-Attn outperforms fine-tuning with S^2-Attn and SW-Attn on natural language tasks.** We fine-tune Llama1 with a context size of 16384 using various Attention variants. Similar to applying SE-Attn to Mamba-2-Hybrid, here we again observe that fine-tuning Llama with SE-Attn improves upon SW-Attn and S^2-Attn.

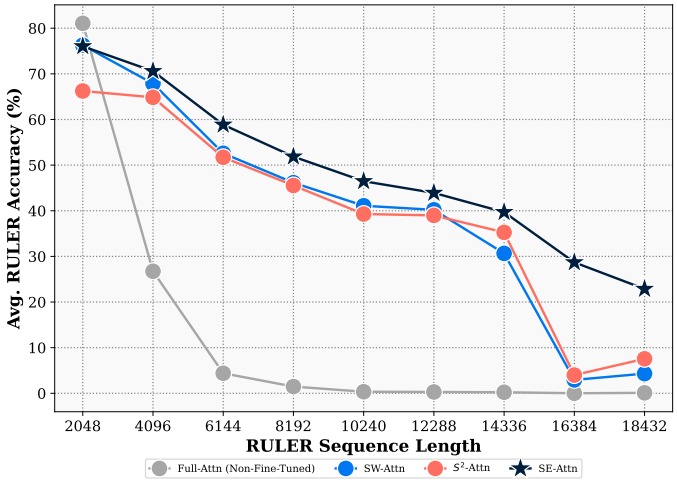


Figure 7: **Fine-tuning with SE-Attn outperforms SW-Attn and S^2-Attn on the RULER benchmark when applied to Llama1.** We fine-tune Llama1 with a context size of 16384 using various Attention variants. We average over eleven RULER tasks, as explained in Appendix I.4. Fine-tuning with SE-Attn consistently outperforms SW-Attn and S^2-Attn even when evaluating on context sizes beyond the fine-tuning size.

LM Harness. Fine-tuning Llama with SE-Attn yields a stronger performance on long-context tasks on the LM Evaluation Harness benchmark. As shown in Table 4, fine-tuning with SW-Attn, S^2-Attn, and SE-Attn all improve upon the non-fine-tuned model on shorter context tasks and perform similarly. However, SE-Attn gives a greater performance on long context tasks.

RULER. Fine-tuning Llama with SE-Attn produces a model with stronger performance on RULER tasks than fine-tuning with SW-Attn and S^2-Attn, as illustrated in Figure 7. On average, SW-Attn and S^2-Attn perform similarly, however, fine-tuning with SE-Attn improves performance by $\sim$5%. Despite all models having a similar PPL up to a context size of 16k as shown in Table 4, we observe a substantial difference between the RULER performance of models fine-tuned with our SE-Attn and those fine-tuned with the SW-Attn and S^2-Attn, again suggesting that PPL is not the most accurate assessment of long-context performance, as discussed in Appendix B.

Appendix D. Beyond Mamba-2-Hybrid

In this section, we apply SE-Attn to the Zamba2 1.2B model Glorioso et al. (2024a). Zamba2 uses shared Attention blocks with different LoRA adapters for the shared blocks. Despite its intricate architecture, we can replace these layers with SE-Attn and fine-tune it to perform on larger context sizes. The pre-trained model was trained with a context size of 4096 and we are able to efficiently fine-tune it with a context size of 12,288. We benchmark our model on NIAH tasks from RULER and provide results in Table 5. We compare to Yang et al. (2025) and Yang et al. (2024b) and demonstrate competitive performance to SOTA models.

	S-NIAH-1				S-NIAH-2				S-NIAH-3			
Model	1K	2K	4K	8K	1K	2K	4K	8K	1K	2K	4K	8K
DeltaNet	97.4	96.8	99.0	98.8	98.4	45.6	18.6	14.4	85.2	47.0	22.4	-
Mamba2	99.2	98.8	65.4	30.4	99.4	98.8	56.2	17.0	64.4	47.6	4.6	-
Gated DeltaNet	98.4	88.4	91.4	91.8	100.0	99.8	92.2	29.6	86.6	84.2	27.6	-
Zamba2-Hybrid	100.0	99.2	99.4	0.0	100.0	100.0	75.0	0.0	94.4	75.4	37.6	0.0
Zamba2-Hybrid + SE-Attn	100.0	100.0	100.0	41.0	100.0	100.0	100.0	42.0	100.0	100.0	94.0	34.0

Table 5: **SE-Attn can efficiently extend the context length of Zamba2-Hybrid**. We fine-tune Zamab2-Hybrid using SE-Attn and evaluate on RULER NIAH tasks.

Appendix E. Additional Fine-Tuning Results on Mamba-2-Hybrid

In this section, we provide additional benchmarks on the Mamba-2-Hybrid model. We begin by expanding upon Figure 2(a) with additional RULER tasks in Figure 8. Here, we observe that Mamba-2-Hybrid fine-tuned with SE-Attn consistently outperforms SW-Attn and S^2-Attn on all RULER tasks across a broad range of context sizes—even beyond the 8192 fine-tuning context size.

E.1. Fine-Tuning on Natural Language + Code

Next, we consider fine-tuning on a dataset that augments natural language with a code dataset. In particular, we construct a dataset consisting of 70% natural language, and 30% C++ code extracted from the Lots of Code Zavadskyy (2017) dataset. We provide PG-19 validation PPL results, as well as results on tasks from the LM Evaluation Harness suite in Table 6. Similar to fine-tuning on natural language only (as in Table 1), we observe that fine-tuning with SE-Attn yields the strongest performance on downstream tasks. We provide performance on RULER in Figure 9. We again observe that fine-tuning with SE-Attn yields the strongest perfromance compared to other efficient Attention layers. Moreover, compared to fine-tuning only on natural language, here we observe a greater improvement on the Variable Tracking task. As explained in Appendix I.4, this task requires the model to keep track of the values of variables that are defined and overridden throughout the input context, and must then return the variables equal to some value. This requires strong recall capabilities, which fine-tuning with SE-Attn enables, and is amplified by the use of code data in the fine-tuning data mix.

E.2. Fine-Tuning on PG-19

Next, we consider fine-tuning on the PG-19 Rae et al. (2019b) dataset. We provide perplexity results on the validation set, along with LM Harness task results in Table 7. Compared to fine-tuning on a different natural language dataset, and a natural language + code dataset as in Table 1 and Table 6,

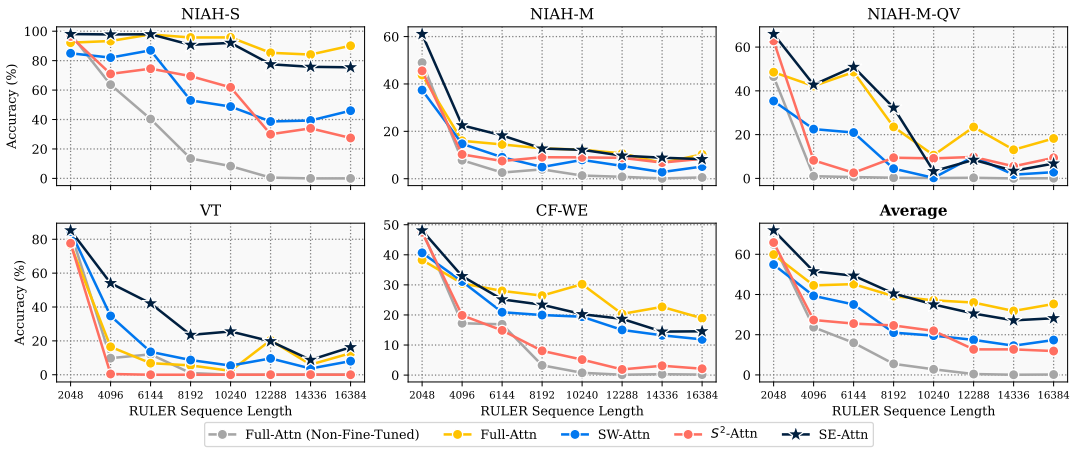


Figure 8: **Mamba-2-Hybrid RULER benchmark fine-tuned on natural language.** We fine-tune Mamba-2-Hybrid with a context size of 8192 using multiple Attention variants and evaluate on the RULER benchmark (see Appendix I.4 for definitions of each metric). On average, fine-tuning with SE-Attn yields a performance close to the paragon model fine-tuned with Full-Attn . We observe the biggest gain in needle-in-a-haystack (NIAH) tasks, as well as variable tracking (VT), all of which require strong recall capabilities enabled by our SE-Attn.

Attention	Eval Context Size (PG-19 PPL ↓)				Short Context Tasks (↑)							Long Context Tasks (↑)			
	2048	8192	16384	32768	ARC-E	ARC-C	Hella.	LAMB.	PIQA	WG	Avg.	SWDE	SQA	SNQA	Avg.
Non-fine-tuned	10.72	14.99	19.35	26.37	69.91	37.97	67.62	69.84	76.06	65.04	64.41	85.60	15.18	3.65	34.81
Full-Attn	10.94	10.23	10.36	11.09	70.12	38.14	67.40	69.34	75.08	64.56	64.11	84.88	25.99	19.61	43.49
SW-Attn	10.94	10.76	11.81	13.43	69.70	38.82	67.51	69.38	75.41	64.88	64.28	84.52	24.44	15.30	41.42
S^2-Attn	10.86	12.89	14.67	16.36	69.95	37.88	67.45	69.94	76.01	64.72	64.32	86.50	16.65	8.61	37.25
SE-Attn	10.95	10.41	11.07	12.50	70.45	38.91	67.39	69.07	75.08	64.96	64.31	85.24	26.14	18.08	43.15

Table 6: **Fine-tuning Mamba-2-Hybrid with SE-Attn on a natural language + code dataset outperforms fine-tuning with S^2-Attn and SW-Attn on natural language tasks.** We fine-tune Mamba-2-Hybrid with a context size of 8192 using various Attention variants on a dataset that consists of 70% natural language and 30% code. We evaluate PG-19 validation perplexity (PPL) and observe that fine-tuning with SE-Attn yields better perplexity scores than S^2-Attn and SW-Attn. On short-context tasks from the LM Harness suite, all models perform similarly. On long context tasks from the LM Harness suite, SE-Attn outperforms S^2-Attn and SW-Attn.

here we obtain lower perplexity scores due to the lack of distribution shift. Interestingly, compared to fine-tuning on the previously mentioned datasets, we observe a slight degradation on LM Harness tasks. Moreover, in Figure 10 we plot RULER performance when fine-tuning on PG-19, and here we also see a decay in performance compared to fine-tuning on the previous datasets. This suggests that the PG-19 dataset may be too far out of distribution for these long-context retrieval task. Nevertheless, we see that fine-tuning with SE-Attn yields the best performance across all of these tasks.

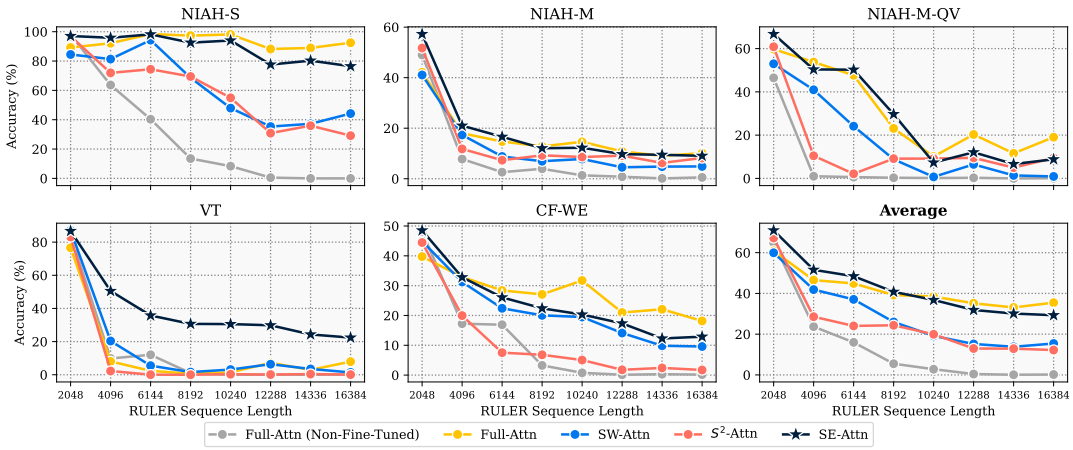


Figure 9: **Mamba-2-Hybrid RULER benchmark fine-tuned on natural language + code.** We fine-tune Mamba-2-Hybrid with different Attention layers on a dataset that consists of 70% natural language and 30% code and evaluate on RULER. Compared to fine-tuning on only natural language (as in Figure 8), we see a substantial improvement on tasks like variable tracking (VT), and needle-in-a-haystack tasks (NIAH), both of which require strong recall capabilities enabled by fine-tuning with SE-Attn.

Attention	Eval Context Size (PG-19 PPL ↓)				Short Context Tasks (↑)							Long Context Tasks (↑)			
	2048	8192	16384	32768	ARC-E	ARC-C	Hella.	LAMB.	PIQA	WG	Avg.	SWDE	SQA	SNQA	Avg.
Non-fine-tuned	10.72	14.99	19.35	26.37	69.91	37.97	67.62	69.84	76.06	65.04	64.41	85.60	15.18	3.65	34.81
Full-Attn	10.73	10.04	10.14	10.91	67.34	37.12	66.80	68.62	74.32	62.67	62.81	84.88	25.35	18.46	42.90
SW-Attn	10.72	10.59	11.64	13.25	66.96	37.54	66.83	68.79	74.54	63.30	62.99	84.79	22.54	14.09	40.48
S^2-Attn	10.78	12.74	14.42	16.11	69.36	38.14	67.45	69.90	76.12	64.72	64.28	86.50	17.00	7.74	37.08
SE-Attn	10.73	10.22	10.84	12.28	67.42	37.88	66.48	69.22	73.88	61.88	62.80	85.15	23.06	16.46	41.55

Table 7: **Fine-tuning Mamba-2-Hybrid with SE-Attn on PG-19 outperforms fine-tuning with S^2-Attn and SW-Attn on long-context natural language tasks.** We fine-tune Mamba-2-Hybrid with a context size of 8192 using various Attention variants on PG-19. We evaluate PG-19 validation perplexity (PPL) and observe that fine-tuning with SE-Attn yields better perplexity scores than S^2-Attn and SW-Attn. On short-context tasks from the LM Harness suite, S^2-Attn has the strongest performance. On long context tasks from the LM Harness suite, SE-Attn outperforms S^2-Attn and SW-Attn.

E.3. Additional Long-context Benchmarks

In this section, we evaluate our Mamba-2-Hybrid model fine-tuned on only language data on additional long-context tasks. First, we test our models' long-context understanding on the LongBench benchmark and report results in Table 8. Here, we observe that our fine-tuned models improve over the pre-trained baseline models, suggesting that these models' long-context understanding has improved. Moreover, we see that models fine-tuned with our SE-Attn match the performance of the model fine-tuned with Full-Attn .

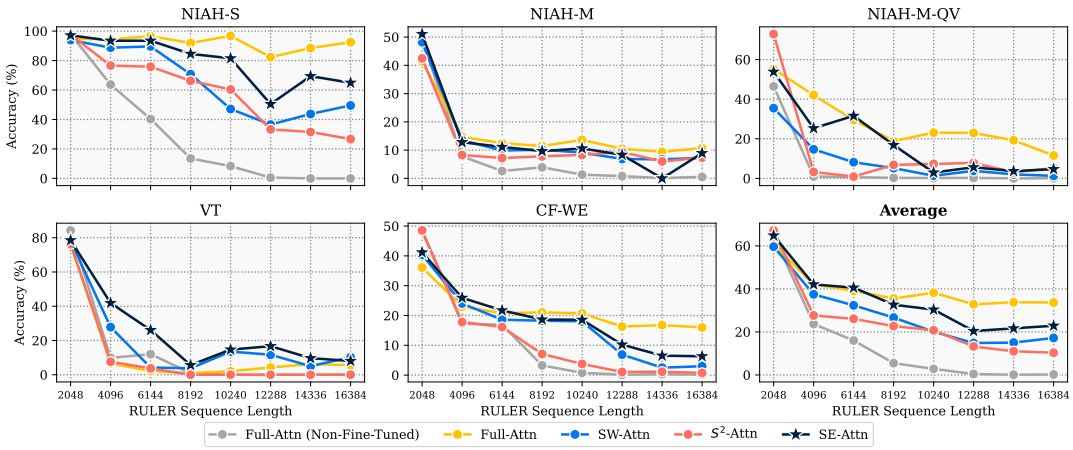


Figure 10: **Mamba-2-Hybrid RULER benchmark fine-tuned on PG-19.** We fine-tune Mamba-2-Hybrid with different Attention layers on the PG-19 Rae et al. (2019b) dataset and then evaluate on RULER. Compared to fine-tuning on other natural language datasets with a greater variety of text as in Figure 8, here we observe a degradation in performance across all models, likely due to a distribution shift in the PG-19 data and the RULER tasks.

Attention	Single-Doc QA (↑)			Multi-Doc QA (↑)			Summarization (↑)			Few-shot (↑)		Code (↑)		
	NQA	QQA	MFQ	HQA	2WM	Mus	GvR	QMS	MNs	TRC	TQA	LCC	RBP	Avg.
Non-fine-tuned	0.92	3.78	13.92	4.72	8.26	1.54	9.65	3.71	23.23	59.0	63.22	60.55	31.99	21.88
Full-Attn	3.25	7.15	16.35	8.03	9.51	3.69	25.4	17.64	26.05	64.5	82.09	56.67	55.58	28.92
SW-Attn	2.29	7.52	17.07	7.72	9.4	3.41	18.44	11.87	23.94	64.5	81.48	58.33	53.06	27.62
S^2-Attn	1.39	5.58	16.43	5.31	7.88	1.83	13.64	5.64	26.86	57.0	72.97	56.62	36.87	23.69
SE-Attn	3.19	9.37	16.0	8.31	9.31	3.94	24.65	14.39	26.86	66.5	80.82	58.71	53.65	28.90

Table 8: **SE-Attn matches the performance of Full-Attn on long-context understanding tasks**. We fine-tune Mamba-2-Hybrid models using various attention mechanisms on 14 LongBench Bai et al. (2024) tasks.

We previously considered RULER as a means of assessing our models' in-context recall capabilities. While RULER can assess our models' performance on synthetic in-context recall tasks, we also want to evaluate our their performance on "real-world" in-context tasks. As such, in Table 9, we evaluate on the tasks considered in Arora et al. (2024). We observe that fine-tuning with our SE-Attn is able to match the performance of fine-tuning with Full-Attn on these tasks.

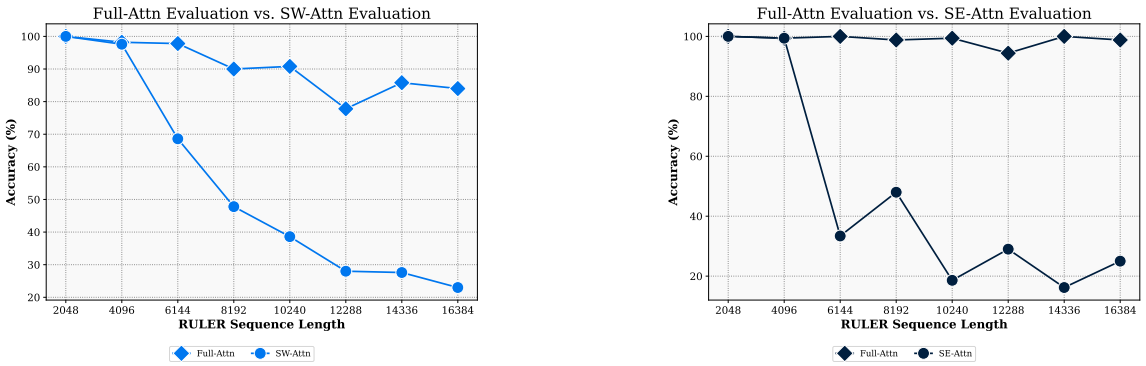
E.4. Evaluating with Efficient Attention

In Appendix A, we explained how we fine-tune our models using an efficient Attention mechanism, and then evaluate with full attention. In Table 2, we showed that evaluating our models with efficient Attention mechanisms produced strong results on perplexity benchmarks. However, we did not observe similarly positive results on more complex tasks, such as those in RULER. As illustrated in Figure 11, using the same efficient Attention layer used during fine-tuning does not perform as well as evaluating with Full-Attn (we omit results for models fine-tuned with S^2-Attn and evaluated with

Attention	SWDE	SQUADv2	FDA	TriviaQA	NQ	Drop	Avg
Non-fine-tuned	85.60	50.11	76.41	22.60	6.62	3.20	40.76
Full-Attn	85.24	50.09	80.22	25.75	6.79	3.55	41.94
SW-Attn	84.61	50.10	78.31	25.51	6.76	3.62	41.48
S^2-Attn	86.41	50.11	81.49	23.63	6.84	3.32	41.97
SE-Attn	85.96	50.09	82.12	26.39	7.17	3.76	42.58

Table 9: **SE-Attn matches the performance of Full-Attn on real-world in-context tasks**. We fine-tune Mamba-2-Hybrid models with different attention layers. We do not truncate inputs to 2K tokens when evaluating as was done in Arora et al. (2024).

S^2-Attn as the implementation of S^2-Attn does not support evaluating with it at arbitrary sequence lengths). Hence, our training pipeline involves training a model with an efficient Attention layer, and reverting to Full-Attn during inference. Moreover, we again note that perplexity is a misleading metric in regard to measuring a model's performance on long context tasks. Though we observed strong performance on perplexity when evaluating with these efficient Attention layers, this did not translate to stronger performance on tasks that require a strong recall capability, such as those in RULER.



(*a*) (*b*)

Figure 11: **Evaluating with efficient Attention mechanisms on RULER does not do as well as evaluating with standard full Attention.** We fine-tune Mamba-2-Hybrid with SW-Attn (a) and SE-Attn (b) and then evaluate on the NIAH-Single-1 RULER task using either the same Attention layer used for fine-tuning, or Full-Attn. We observe that fine-tuning with efficient Attention layers (SE-Attn and SW-Attn) and then using Full-Attn during evaluation yields better results. Lines with a circle marker denote models fine-tuned with an efficient Attention mechanism, and then evaluated with the same Attention mechanism; lines with a diamond marker were fine-tuned with an efficient Attention mechanism, but evaluated with Full-Attn. We omit results for evaluating models fine-tuned with S^2-Attn as this Attention mechanism does not support arbitrary sequence lengths during inference.

Appendix F. Retrieval with Landmark Tokens

Landmark Attention Mohtashami and Jaggi (2023) was recently introduced as a way for Transformer models to process long sequences by inserting "landmark" tokens into the sequence whose representations would then be used as summaries of the blocks of tokens that came before them. At a high level, our approach in SE-Attn is similar. However, we simplify the process of compressing blocks of tokens by forgoing the use of landmark tokens and instead using Attention to summarize them, as described in Section 3.3. Moreover, to learn which blocks to retrieve, we do not rely on a complex Grouped Softmax function, and instead use a simple Cross-Attention score to ascertain relevance. In this way, we implement retrieval natively into the model's architecture.

In this section, we consider a variant of SE-Attn, which we refer to as SE-Attn-LM, that is inspired by Landmark Attention. Namely, instead of using our Attention-based compression to construct summaries of memory blocks, we insert a non-learnable "landmark" token into each memory block, and use the cross-attention between this token and the memory tokens as the summary of the memory block. We compare the performance of this variant to our summaries computed using average pooling of Attention scores (see Section 3.3) in Figure 12. Here, we see that SE-Attn yields better performance. We suspect this is because using a non-learnable landmark token to summarize memory blocks is too challenging of a task to accomplish using standard LoRA. While full fine-tuning (without LoRA) may improve the performance of SE-Attn-LM, this is beyond the scope of our work as we prioritize efficiency.

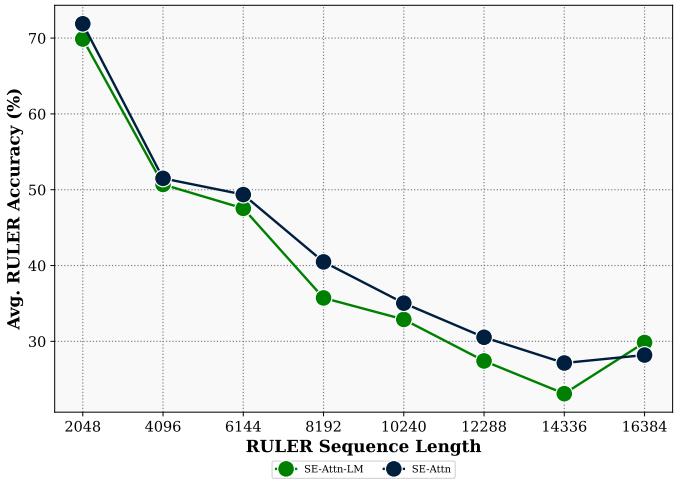


Figure 12: **Summarizing memory blocks via average pooling of attention yields stronger performance than summarizing them using "landmark" tokens.** We fine-tune Mamba-2-Hybrid using our SE-Attn, and SE-Attn-LM, which summarizes memory blocks with a landmark token (similar to Mohtashami and Jaggi (2023)) instead of using the average of the Self Attention output of memory blocks, as in SE-Attn. We find that our simpler SE-Attn produces a stronger model, likely due to the easier training task, which does not require adapting the model to leverage landmark tokens for compression.

Appendix G. Training Details

Our fine-tuning recipe largely follows Chen et al. (2024), with the exception of using a larger learning rate for SE-Attn, SW-Attn, and Full-Attn models (2×10^{-4} vs. the default 2×10^{-5} for S^2). We found that using a larger learning rate for S^2 did not improve its performance, as shown in Figure 13, so we used the default 2×10^{-5} learning rate for all S^2 fine-tuning experiments.

All of our experiments are conducted on a single 8xA100 node. We fine-tune for 1000 steps on a total of 0.5B tokens. We fine-tune Mamba-2-H with a context size of 8192 with 8 accumulation steps and a per-device batch size of 1. We fine-tune Llama1 with with a context size of 16384 tokens with 4 accumulation steps and a per-device batch size of 1. We use FlashAttention-2 Dao (2024) and DeepSpeed Stage 2 Rasley et al. (2020).

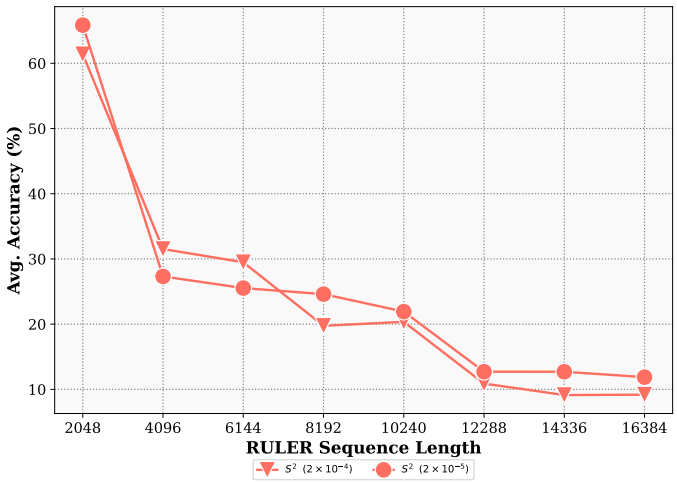


Figure 13: **A larger learning rate does not improve the performance of S^2.** Here we fine-tune a Mamba-2-Hybrid model using S^2-Attn with two different learning rates: 2×10^{-4} and 2×10^{-5}. The learning rate used in Chen et al. (2024) is 2×10^{-5}, which we found to work well. For all other Attention layers, we found 2×10^{-4} to offer a slight improvement over 2×10^{-5}.

Appendix H. Runtime and Memory Analysis

The runtime of SE-Attn is lower than Full-Attn and S^2-Attn on large contexts. For an input with sequence length L, SE-Attn first constructs $U = \frac{L}{S}$ memory blocks of size S. Attention is applied on each. Thus, the complexity of this operation is $O(d_{\text{model}} U S^2) = O(d_{\text{model}} L S)$. Next, the input sequence is split into $T = \frac{L}{M}$ chunks of size M. Cross-Attention is then applied between each chunk's query and each memory block's compressed representation; the cost of this is $O(d_{\text{model}} MTU) = O(d_{\text{model}} \frac{L^2}{S})$. Cross-Attention is then applied between each chunk's query tokens and the concatenations of the chunk's key tokens and memory tokens. Since SE-Attn retrieves K blocks with S tokens in each, the cost of this Cross-Attention for each chunk is $O(d_{\text{model}} M(SK + M)) = O(d_{\text{model}} M^2)$ since $SK < M$. The cost for all chunks is therefore

$O(d_{\text{model}}TM^2) = O(d_{\text{model}}LM)$. The total cost of SE-Attn is therefore $O(d_{\text{model}}LS + d_{\text{model}}LM + d_{\text{model}}\frac{L^2}{S})$. For sufficiently large S, the runtime of SE-Attn is faster than Full-Attn and S^2-Attn, especially on large contexts, and is similar to that SW-Attn.

Appendix I. RULER Task Definitions

The RULER benchmark Hsieh et al. (2024) consists of four different task categories: retrieval, multi-hop tracing, aggregation, and question answering. In this paper, we focus only on the retrieval, multi-hop tracing, and aggregation tasks (we provide question answering results on the LM Evaluation Harness benchmark). These three categories span eleven different tasks as explained below.

I.1. Needle-in-a-Haystack (NIAH) Tasks

RULER consists of 8 different NIAH tasks. These tasks embed "needles" in a string of noise. These needles are typically key-value pairs, and the goal is to return the value of a key. These tasks are characterized by six parameters:

- type_haystack (TH): This specifies the type of noise to embed the key in. The choices are "repeat" which constructs noise as in Mohtashami and Jaggi (2023), "essay" which will use sentences from the Paul Graham essays Kamradt (2023), or "needle" in which case each sentence will define a new key-value pair.
- type_needle_k (TK): This specifies the type of the needle's key. The options are "words", in which case the key is a word (in the form of adjective-noun, e.g., spiritual-oven), or "uuids" in which case the key is a UUID.
- type_needle_v (TV): This specifies the type of the needle's value. It can either be "numbers" in which case the value is a 7-digit number, or it can be "uuids" in which case the value is a UUID.
- num_needle_k (NK): This specifies the number of key-value pairs to embed in the haystack.
- num_needle_v (NV): This specifies how many different values a key is assigned. If greater than 1, the goal is output all the values of they key.
- num_needle_q (NQ): This specifies the number of different keys the model must return the value for.

I.2. Multi-hop Tracing Tasks

RULER considers a "variable tracking" task that is a form of coreference resolution. In this task, a sequence of variables are defined throughout noisy text as in Mohtashami and Jaggi (2023). New variables are defined as previous ones, and a final value is assigned to a particular variable. The goal is to be able to trace back which variables have also been assigned the final value, i.e., determine which variables refer to the final value. We use the default num_chains=1 and num_hops=4 parameters

NIAH Task	TH	TK	TV	NK	NV	NQ
Single 1	repeat	words	numbers	1	1	1
Single 2	essay	words	numbers	1	1	1
Single 3	essay	words	uuids	1	1	1
Multikey 1	essay	words	numbers	4	1	1
Multikey 2	needle	words	numbers	1	1	1
Multikey 3	needle	uuids	uuids	1	1	1
Multivalue	essay	words	numbers	1	4	1
Multiquery	essay	words	numbers	1	1	4

Table 10: **RULER NIAH definitions.** The "Needle-in-a-Haystack" (NIAH) tasks in the RULER benchmarks are defined by 6 parameters which modulate the difficulty of the tasks. We consider 8 different NIAH tasks as defined above (these are the default NIAH tasks in the RULER library).

I.3. Aggregation Tasks

RULER considers two aggregation tasks, common words extraction (CWE), and frequent words extraction (FWE). In CWE, the context consists of list of words, and the goal is to return the most common words. We use the default parameters freq_cw=30, freq_ucw=3, and num_cw=10. In FWE, the context consists of random word strings, and the goal is to return the ones that appear the most frequently. We use the default alpha=2 parameter for this.

I.4. Aggregating RULER Tasks

We aggregate the eleven RULER tasks above into six groups:

1. *NIAH-S*: NIAH Single 1, NIAH Single 2, NIAH Single 3.
2. *NIAH-M*: NIAH Multikey 1, NIAH Multikey 2, NIAH Multikey 3.
3. *NIAH-M-QV*: NIAH Multivalue, NIAH Multiquery.
4. *VT*: Variable Tracking.
5. *CF-WE*: Common Words Extraction (CWE) and Frequent Words Extraction (FWE).
6. *Average*: The average of all eleven tasks above.

Appendix J. Task Abbreviations

LM Harness: Hella=HellaSwag, LAMB=LAMBADA, WG=WinoGrande, SQA=ScrollsQAsper, SNQA=ScorllsNarrativeQA.
LongBench: NQA=NarrativeQA, QQA=QasperQA, MFQ=MultiFieldQA, HQA=HotpotQA, 2WM=2WikiMultihopQA, Mus=Musiqe, GvR=GovReport, QMS=QMSum, MNs=MultiNews, TRC=TREC, TQA=TriviaQA, SSM=SamSum, RBP=RepoBench-P.