SHORT WINDOW ATTENTION ENABLES LONG-TERM MEMORIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent works show that hybrid architectures combining sliding window softmax attention layers with linear recurrent neural network (RNN) layers outperform both of these architectures taken separately. However, the impact of the window length and the interplay between softmax attention and linear RNN layers remain under-studied. In this work, we introduce SWAX, a hybrid architecture consisting of sliding-window attention and xLSTM linear RNN layers.

A counter-intuitive finding with SWAX is that larger sliding windows do not improve the long-context performance. In fact, short window attention encourages the model to better train the long-term memory of the xLSTM, by relying less on the softmax attention mechanism for long context-retrieval.

The issue with small sliding windows is that they are detrimental for short-context tasks, which could be solved with information from moderately larger sliding windows otherwise. Therefore, we train SWAX by stochastically changing the sliding window size, forcing the model to leverage both a longer context window and the xLSTM memory. SWAX trained with stochastic window sizes significantly outperforms regular window attention both on short and long-context problems.

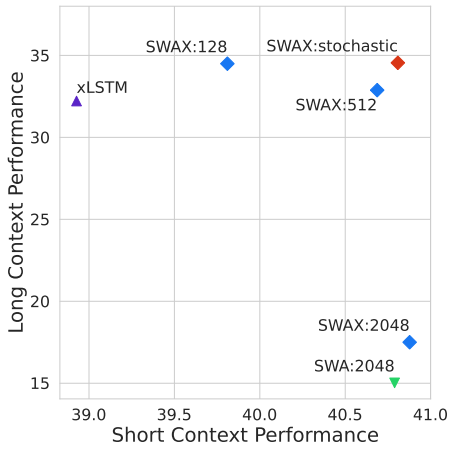

Figure 1: Short (average score across benchmarks) vs long context performance for 1.4B xLSTM, SWA (sliding window attention) and SWAX with different sliding window sizes.

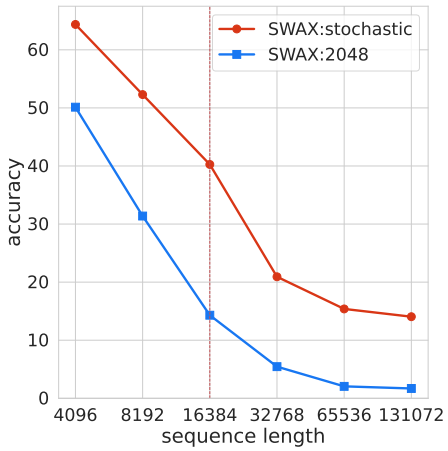

Figure 2: RULER Needle-In-A-Haystack accuracy of a 1.4B SWAX model with a fixed sliding window size of 2048 vs our method using a stochastic window size of 128/2048.

1 INTRODUCTION

Memory is a core concept in Neural Network Architectures (Zhong et al., 2025). Modern LLMs based on softmax attention have a working memory in the form of the key-value (KV) Cache and yield state-of-the-art long-context performance. This working memory expands indefinitely as the sequence length grows, incurring a linear growth in both compute and memory to generate each new token. With such an unbounded compute cost, current models become prohibitively expensive for in-context learning on long sequences such as codebases and long reasoning traces.

On the other hand, recurrent neural networks (RNNs) like State Space Models (SSMs) (Gu & Dao, 2024) or variants of linear attention (LA) (Katharopoulos et al., 2020) maintain and iteratively update a hidden state. Through input-dependent update rules, RNNs manage to decide whether to keep previous or add new information (Hochreiter & Schmidhuber, 1997; Chung et al., 2014). In this way, the compute and memory cost is constant and independent of the sequence length. This allows models to learn at test time from large sequence lengths and reason without a specific token limit. Recently, these linear RNNs have been generalized in the context of online learning (Liu et al., 2024; Sun et al., 2025). However, the recall ability of linear RNNs remains inferior to that of Transformers (Fu et al., 2023). This shortfall has hindered their adoption in favor of global attention-based architectures, which remain the current state-of-the-art architecture for language and code models (Jain et al., 2024).

Recent works like those of De et al. (2024), Ren et al. (2025), Dong et al. (2024) and Arora et al. (2025) have aimed at combining the advantages of softmax attention and Linear Attention into hybrid architectures (Wang et al., 2025). Following this line of research, in this paper, we study hybrid architectures, which combine linear RNNs and *sliding window* attention – both components with fixed maximum state size and thus fixed compute cost per token.

In this context, we make the following contributions:

1. We investigate the impact of the sliding window length on a wide range of tasks, encompassing validation perplexity, short-context reasoning, common sense benchmarks, and long-context modeling tasks;
2. We show that, contrary to previous belief, for hybrids architecture interleaving sliding window attention and linear RNNs, *longer* sliding windows actually *hurt* performance in long-context retrieval tasks compared to using *shorter* windows;
3. We present a training strategy based on a stochastic window size that provides the long-context performance enabled by short windows and the short-context and reasoning performance of longer windows.

2 Background

The attention mechanism handles sequences of key vector $\mathbf{k}_t \in \mathbb{R}^{d_{qk}}$ and value vectors $\mathbf{v}_t \in \mathbb{R}^{d_v}$. A fundamental perspective proposed by Katharopoulos et al. (2020) is that all forms of attention update a matrix memory by adding to it the outer product of key vector $\mathbf{k}_t \in \mathbb{R}^{d_{qk}}$ and value vectors $\mathbf{v}_t \in \mathbb{R}^{d_v}$. This holds, provided that we can apply a vector mapping to each of these vectors. Then, to read from the memory, a query $\mathbf{q}_t \in \mathbb{R}^{d_{qk}}$ is compared to the previous keys using a similarity metric, usually the inner product $\langle \mathbf{q}, \mathbf{k} \rangle$. In order to improve the accuracy of subsequent retrieval operations, a pre-processing feature mapping ϕ is applied to the keys and queries. Defining the memory tensor as

$$\mathbf{H}_t = \sum_{t=1}^{S} \phi(\mathbf{k}_t)\, \mathbf{v}_t^\top \in \mathbb{R}^{d_{qk} \times d_v}, \qquad \mathbf{z}_t = \sum_{t=1}^{S} \phi(\mathbf{k}_t) \in \mathbb{R}^{d_{qk}}, \tag{1}$$

a normalized read is performed as

$$\mathbf{y}_t = \frac{\phi(\mathbf{q}_t)^\top \mathbf{H}_t}{\phi(\mathbf{q}_t)^\top \mathbf{z}_t} = \frac{\sum_{i \le t} \langle \phi(\mathbf{q}_t), \phi(\mathbf{k}_i) \rangle\, \mathbf{v}_i}{\sum_{i \le t} \langle \phi(\mathbf{q}_t), \phi(\mathbf{k}_i) \rangle}. \tag{2}$$

Linear Attention. Equation 1 shows that if the kernel if ϕ is a finite-dimensional mapping, then the feature-mapped keys as well as the memory tensor are also finite-dimensional and can be materialized and cached for future retrievals $(\mathbf{H}_t, \mathbf{z}_t)$ in *constant* memory:

$$\mathbf{H}_t \leftarrow \mathbf{H}_{t-1} + \phi(\mathbf{k}_t)\, \mathbf{v}_t^\top, \qquad \mathbf{z}_t \leftarrow \mathbf{z}_{t-1} + \phi(\mathbf{k}_t). \tag{3}$$

All the keys and values are thus stored in constant memory. The per-token read cost is $O(d_{qk} \times d_v)$. Importantly, it does not depend on the sequence length S.

Softmax Attention (SA). Katharopoulos et al. (2020) show that softmax attention can be seen as performing the attention operation defined in Equations 1 and 2, i.e., as writing outer products between keys and values in a memory. In such a case, the keys and queries undergo an infinite-dimensional feature mapping induced by the exponential kernel in softmax attention. Compared to linear attention, an infinite-dimensional exponential feature map reduces the interference between the stored keys and yields an improved retrieval accuracy. Another consequence is that the memory $\mathbf{H}_t$ cannot be materialized and cached. Instead, one needs to maintain *all* the previous keys and queries in memory in order to compute the exponential of the dot products of the keys and queries, also referred to as the "KV Cache". A well known issue inherent to the self-attention mechanism, is that the KV Cache size (and per token computation) increases linearly with the sequence length.

Gated Linear Attention. Another way to limit interference between keys in the sequence is to learn when to "forget" information and remove it from the memory. This is the idea behind Gated Linear Attention (Yang et al., 2024), which improves stability and long-context performance through selective retention/forgetting of the information. Let $\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \boldsymbol{\lambda}_t \in \mathbb{R}^{d_{qk}}$ be write-, read-, and decay-gates or, equivalently, broadcastable vectors. Gating is often implemented by learned affine maps and element-wise sigmoids. The update and reading rule are as follows:

$$\mathbf{H}_t = \operatorname{diag}(\boldsymbol{\lambda}_t)\, \mathbf{H}_{t-1} + \operatorname{diag}(\boldsymbol{\alpha}_t)\, \boldsymbol{\phi}(\mathbf{k}_t)\, \mathbf{v}_t^\top, \tag{4}$$

$$\mathbf{y}_t = \left(\operatorname{diag}(\boldsymbol{\beta}_t)\, \boldsymbol{\phi}(\mathbf{q}_t) \right)^\top \mathbf{H}_t. \tag{5}$$

Gated Linear Attention as well as other modern RNNs remove the normalizing constant and, instead, rely on normalizing layers such as LayerNorm (Ba et al., 2016) and RMSNorm (Zhang & Sennrich, 2019) in the network to stabilize training (Beck et al., 2025a).

Sliding Window Attention (SWA). Softmax attention maintains all past (k_i, v_i) pairs, producing linear growth in memory and compute with t due to the KV cache. Variants with a sliding window of size w restrict the attention process to only the previous w tokens, changing the memory and time complexity per-token from $O(S)$ to $O(w)$ complexity (Beltagy et al., 2020). This theoretically allows SWA architectures to handle arbitrarily large input sequences. In practice the receptive field of the model is limited to $O(lw)$ where l is the number of SWA layers in the model. Moreover, it is unlikely that the theoretical receptive field is fully utilized in practice (Xiao, 2025).

Hybrids between Local Attention and Global Softmax Attention. Through multi-turn interactions (Gehring et al., 2025), tool-use or long Chain-Of-Thought reasoning (Wei et al., 2023; DeepSeek-AI, 2025), the length which models have to process has grown from a few thousands of tokens to tens or hundreds of thousands of tokens. This motivates several recent works (OpenAI, 2025; Dong et al., 2024; NVIDIA, 2025; Ren et al., 2025) to consider new architectures whose computational cost grow less rapidly relative to sequence length than that of global softmax attention, while still performing well on long-context tasks. One such type of architectures are hybrids, for which most layers have a fixed state size like sliding-window or Linear Attention Layers, and the rest of the layers are global softmax attention layers. However, because those architectures still keep some global attention layers to remain competitive on long-context tasks, they also keep the $O(S)$ scaling in state size and FLOPs per token.

Hybrids between Linear Attention and Sliding Window Softmax Attention. Another kind of hybridization involves only component with a fixed state size, as considered by De et al. (2024) and Ren et al. (2025), who hybridize linear attention variants with sliding window attention. SWA paired with linear attention provides a natural split: the linear path maintains a compressed working memory with an unlimited receptive field; the windowed softmax path offers high-fidelity local reasoning. Moreover, they demonstrate that, despite having *less* layers LA layers which are the only ones with an unlimited receptive field, the long-context performance of such hybrids is actually *higher* than that of a purely Linear Attention architecture. In particular, (De et al., 2024) investigated the impact of the size of the sliding window on validation perplexity. They found that longer windows yield better performance, making the choice of window size a purely a trade-off between performance and compute. However, they did not investigate the impact of the sliding window length on the *long-context* performance of the models.

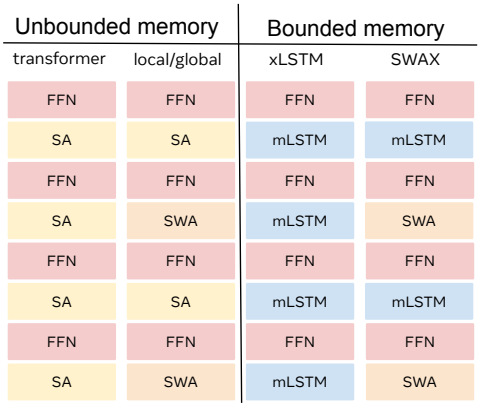

Figure 3: We compare 4 different types of architectures, including 3 hybrid architectures:
(1) The transformer with regular self-attention (SA). Its complexity that becomes prohibitive for long contexts lengths.
(2) This is circumvented by replacing some SA layers by sliding window attention (SWA) layers (Gemma Team, 2025; OpenAI, 2025).
(3) xLSTM (Beck et al., 2024) offers a memory with unbounded time horizon, albeit not as precise as SA for handling the recent context.
(4) SWAX is an hybrid architecture that includes both SWA layers and long-term memories layers, implemented with mLSTM memory cells.

3 Hybrid Architecture Design

In this work, we focus on hybrid architectures that alternate sliding window attention and linear RNNs. As a candidate for the linear RNN component, we choose the xLSTM (Beck et al., 2024), as this architecture has been scaled to models having up to 7B parameters and has shown strong performance in a wide variety of tasks. Importantly, fast and efficient Triton kernels are available (Beck et al., 2025b;a). xLSTM introduced two novel memory cells: the sLSTM with a scalar memory and the mLSTM with matrix memory. However, on language tasks the mLSTM cell shows superior performance over the sLSTM, which has been abandoned in the latest 7B xLSTM model. We follow this choice and rely solely on the mLSTM cell in our hybrid architecture. Subsequently, we use xLSTM and mLSTM interchangeably to refer to the same architecture.

There exist many ways to hybridize these two components (Wang et al., 2025). In our case, we adopt the simple design of inter-layer hybridization, which alternates between SWA layers and layers of linear attention. For the sake of simplicity, we adopt a 1:1 ratio meaning, that for every xLSTM layer there is one Sliding Window Attention layer. Figure 3 illustrates how the layers are interleaved in pure architectures and our SWA-xLSTM hybrid architecture. Most hybrids use window sizes between 128 as in OpenAI (2025) and 2048 as in De et al. (2024). We evaluate at intermediate lengths with sliding attention windows of lengths 128, 256, 512, 1024 and 2048. Finally, we evaluate a stochastic training procedure that aims at improving length-extrapolation. This training procedure stochastically chooses for each new batch either a short or a long window. In our experiments, we sample a window size of either 128 or 2048 with probability 0.5 for each length. A similar strategy was proposed by Zhang & Bottou (2025) in the context of the Memory Mosaic architecture. However, in their case, the stochastic attention mask was applied to a long-term memory layer. In constrast, we apply it to a Sliding Window Attention layer with the explicit goal of reducing over-reliance on the SWA layers for long-context recall.

4 Experiments

4.1 Experimental Setup

Our experiments focus on language modeling, with an emphasis on understanding the compromise between short-context and long-context recall performance. In particular, we investigate the impact of the SWA window size on long-context retrieval. For this purpose, we mainly rely on the needle-in-a-haystack tasks of the RULER benchmark (Hsieh et al., 2024). A common practice for models using global attention is to pre-train them on shorter sequence lengths like 4k or 8k to reduce the cost of the attention operation, and are then fine-tuned in a second training stage on longer sequences to improve their long-context ability (Peng et al., 2023). In our case, we mainly focus on fixed-memory, fixed-compute architectures. Therefore, training longer training sequences does not increase the required compute to attain a total training tokens target. We choose to train our models on 16k sequence length from the start. Since we are interested in the capabilities of the model after standard pre-training, we do not perform any task-specific fine-tuning on long-context tasks. Except stated otherwise, our experiments use a model with 1.4 billion parameters. From our observations, it is

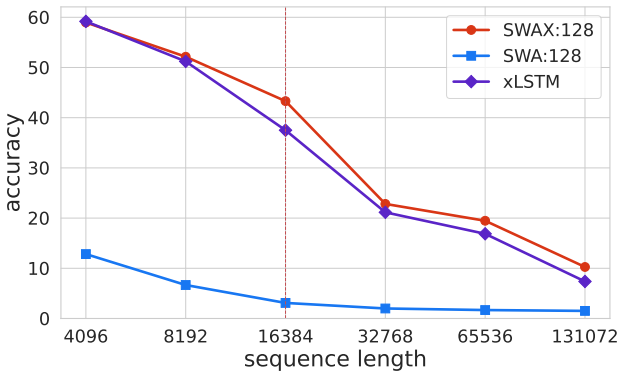

Figure 4: RULER needle in a Haystack average performance on varying sequence lengths for 1.4B models

at this size that models become able to perform recall on sequence lengths in the tens of thousands of tokens. However, to validate our method of stochastic window size at larger scale, we also train evaluate models at the 7B parameter scale. The models have 24 blocks and a model dimension of 2048 while the 7B models have 32 blocks and a model dimension of 4096. In each block, the FFN is a gated MLP Liu et al. (2021) with Silu activation (Elfwing et al., 2017). For the SWA layers of the hybrids, we use Rotary Positional Embedding (RoPE) (Su et al., 2023) with a frequency θ of 10000, and 16 attention heads. All models are trained on 150 billion tokens following a warmup-cosine learning rate schedule with a peak learning rate of $3 \cdot 10^{-4}$ and a minimum learning rate of $3 \cdot 10^{-6}$. The batch size is 10^6 tokens. Our training data mix consists mostly of web-data and code. Since we are most interested in the performance of models on long sequences, and our code data has, on average, 10 times longer documents than our web data, we report the validation perplexity on the code data subset of our data mix. For the MBPP (Austin et al., 2021) and HumanEvalPLus (Liu et al., 2023) pass@10 results, we use a sampling temperature of 0.8.

4.2 HYBRID VS PURE ARCHITECTURES

We start our investigation by reproducing the finding from (De et al., 2024) that hybrids between Local Attention and Linear Attention variants improve performance across the board on both short-term reasoning and long-term recall tasks.

Long-context performance Figure 4 shows that the pure SWA architecture performs poorly on long-context recall. This is expected because of its limited receptive field of 128 * 24 = 3072 tokens. More importantly, it confirms the counterintuitive finding from (De et al., 2024) that hybrids, despite having fewer global receptive field layers, outperform the pure variants in long-context recall. Intuitively, this is explained by the fact that, although the SWA layers have a limited receptive field, the softmax feature mapping allows them to better model local dependencies than an equivalent number of linear attention layers. Since most of the information necessary to predict the next token comes from local dependencies (Ruiz & Gu, 2025), a fully linear attention model dedicates most of its layers to modeling local dependencies and few layers to model long-term dependencies.

On the other hand, in SWA-LA hybrids, local dependencies are rather routed to the softmax local attention layers, which are more precise because of their direct access to recent history. As a consequence the linear attention layers specialize themselves in modeling long-term dependencies, which the SWA layers cannot model due to the limited window size. This highlights the impact that the window size can have on how much supervision the linear attention layers receive.

Short-context performance Table 1 shows that the performance of hybrid models on short-context reasoning benchmarks is higher than that of a xLSTM and also slightly higher than that of a pure SWA architecture. This further highlights the fact that for short contexts, hybrid models leverage the high precision of the softmax sliding window attention layers. Hybrids models therefore take the strong short-context performance of softmax attention, and the improved long-context recall ability of the Linear Attention layers.

Model	Transformer	xLSTM	SWA	SWAX				
window length	n/a	n/a	128	128	256	512	1024	2048
FLOPs/token ($\times 10^9$)	6.174	2.978	3.029	3.004	3.016	3.041	3.092	3.192
val_PPL ↓	**2.431**	2.602	3.036	2.551	2.540	2.546	2.538	**2.523**
HEplus/pass@10 ↑	14.63	12.80	13.41	12.20	12.80	14.02	14.63	**15.24**
ARC-c ↑	30.90	28.93	31.25	29.27	30.13	**31.76**	30.30	29.79
ARC-e ↑	66.43	65.79	65.58	65.75	**67.82**	**67.82**	66.89	67.15
Hellaswag ↑	46.18	44.68	44.45	45.37	45.37	45.47	45.47	**45.91**
MBPP/pass@10 ↑	**31.40**	22.60	23.80	24.20	26.80	28.40	29.20	28.80
NaturalQuestions ↑	13.45	12.49	12.37	**13.51**	13.45	13.40	12.31	13.19
PIQA ↑	73.99	73.39	73.99	73.83	74.05	**74.48**	73.67	73.99
RACE.high ↑	**37.19**	33.65	33.25	33.56	35.71	35.71	35.11	35.99
RACE.mid ↑	**50.63**	46.59	45.54	46.52	49.09	48.89	48.40	49.30
SIQA ↑	42.48	40.23	41.61	**42.53**	42.02	41.71	41.91	41.71
TriviaQA ↑	**30.11**	27.96	28.15	28.95	29.59	28.26	28.98	29.95
Winogrande ↑	61.41	58.01	**62.12**	62.04	59.91	58.33	59.27	59.51
average ↑	**41.57**	38.93	39.63	39.81	40.56	40.69	40.51	**40.88**

Table 1: Validation perplexity and accuracy on short-context reasoning and commonsense tasks. All models have 1.4B parameters. To compute the transformer FLOPs we use the training sequence length of 16384.

4.3 IN SEARCH OF AN OPTIMAL WINDOW SIZE FOR HYBRIDS

Hereafter, we establish that windows that are too long actually hinder the Linear Attention layers from learning to model long-term dependencies during training. In other words, we hypothesize that such long softmax attention windows actually *degrade* the long-context recall abilities of the model when used on longer sequence length than seen during training, due to under-training of the linear attention layers on the long-context recall task. To validate this hypothesis, we train hybrids with a varying window size in 128, 256, 512, 1024, 2048 and test the models on both short context reasoning tasks but, more importantly, also on long-context recall tasks like RULER NIAH.

Short-context performance De et al. (2024) experimented with different window sizes to find the optimal sliding window size. However, they only evaluated the different window sizes using the validation perplexity. Table 1 shows that, indeed, the hybrid with the largest softmax attention window (SWAX:2048) has the best performance from the validation perplexity point of view.

However, raw perplexity is not sufficient to accurately predict performance in downstream tasks, and especially not in long-context modeling tasks (Fang et al., 2024). Thus, we also evaluate the impact of the window size on short-context reasoning and common sense benchmarks and on long-context retrieval tasks from the RULER benchmark. Table 1 shows that on short-context reasoning benchmarks all window sizes except the shortest one, 128, give similar results, with the best performing hybrid being the one trained with the longest window size of 2048. The worse performance of the shortest window of size 128 is understandable as most prompts, even from those relatively short reasoning benchmarks, do not necessarily fit within a sliding window of 128 tokens.

Long context performance Figure 5 shows that once tested on longer sequences, the performance of the hybrid trained with a window size of 2048 drops the most. On the other hand, the SWAX models trained with shorter window sizes like 128, 256, and 512, maintain better performance even up to sequence lengths of 65k and 131k tokens. On the NIAH single task, SWAX models with a shorter window have around 30% recall accuracy at 131k sequence length, while the SWAX with a window of 2048 has near 0% recall. Even the shortest sliding window of size 128, which consistently underperformed the longest ones in terms of PPL and short-context reasoning, significantly outperforms the model with the 2048 window length on all RULER NIAH tasks.

As shown in Figure 6, averaging over all sequence lengths and NIAH tasks, the SWAX model with a window size of 128 actually performs the *best* out of all the window sizes we tested. In particular, it outperforms the 2048 window size by 16 accuracy points. In other words, the SWAX with the shortest window has a recall *88.9% higher* than the SWAX with the longest window. The most likely cause for this phenomenon is that during training, most of the dependencies to model fall inside the 2048 tokens window.

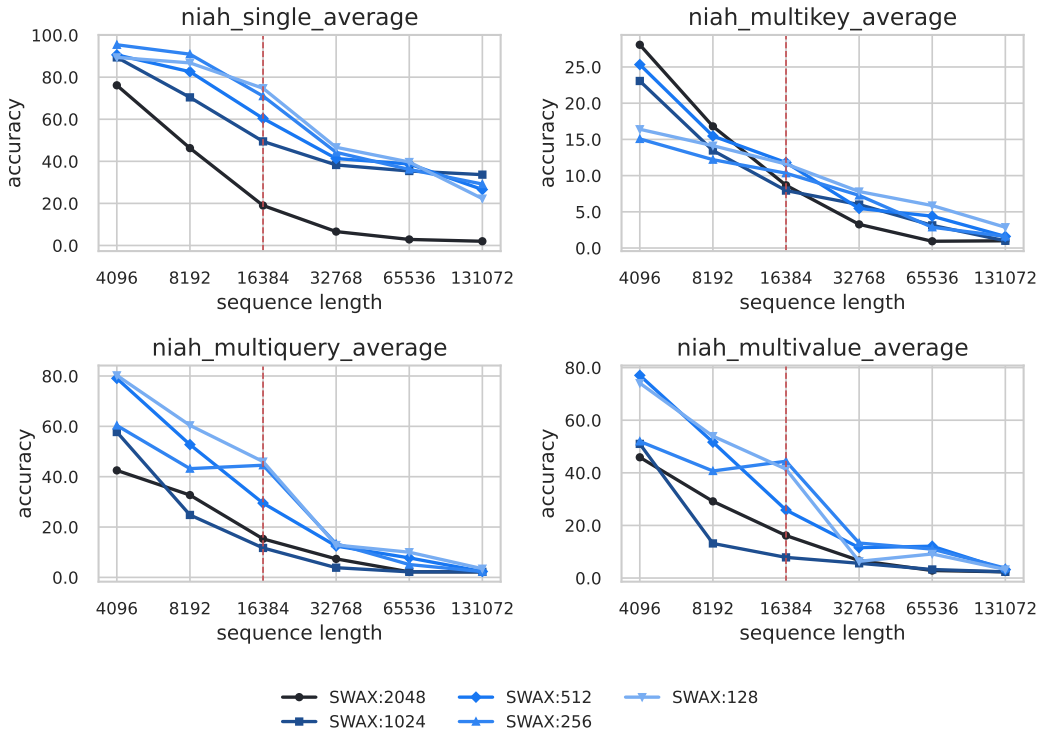

Figure 5: RULER NIAH subtasks accuracy for 1.4B SWAX models with different window sizes

Therefore, during pretraining, it was advantageous for the model with a window of 2048 to use the more precise softmax attention from the sliding window rather than having to rely on the less precise Linear Attention layers to model most dependencies. However, once tested on longer sequence length where the dependencies are outside of the window length, the model does not extrapolate since it never learned to rely on the Linear Attention layers to do long-context modeling.

On the other hand, the models with shorter windows *had to* rely on the Linear Attention layers to propagate information since many dependencies fell outside of the sliding window. We give further evidence in Appendix A which further indicates that this is indeed the reason for the poor long-context performance of hybrids with long sliding windows.

All these results show that, contrary to previous belief, longer sliding windows do not always provide better performance and can even have a strong *negative* impact when extrapolating to tasks beyond the sliding window size and training sequence length. On the contrary, shorter window size push the global receptive field Linear Attention layers to receive more supervisory signal and specialize in long-context dependencies. Overall, shorter sliding windows allow the model to better extrapolate to tasks beyond the sliding window size and even far beyond the training sequence length. This also means that shorter windows are not just a way of reducing computational cost or maximize hardware utilization as was often thought to be the case, as in Arora et al. (2025) and De et al. (2024).

4.4 DIFFERENT WINDOW SIZES AT TRAIN AND TEST TIME

We now explore a training strategy allowing for a large window size at test time, to have the best reasoning performance possible, while still being trained such that the Linear Attention layers for long-term dependencies and extrapolate to longer sequences.

Length extrapolation As a preliminary analysis, we first evaluate the performance of models when tested with a different window size than the one used at training time. Figure 6 shows that, as expected, naively extending the window size beyond its training length results in catastrophic collapse. This is a common phenomenon in softmax attention with RoPE which is used in the

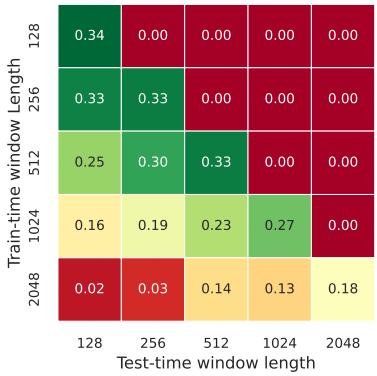

Figure 6: average NIAH accuracy of 1.4B SWAX models depending on their train and test time window sizes.

Table 2: Validation Perplexity and accuracy on downstream tasks. Stochastic models use $w = 2048$ by default and switch to $w = 128$ with probability $p = 0.5$ at 1.4B scale and with probability $p = 0.75$ at 7B scale.

model – parameters	SWAX 1.4B			SWAX 7B		
train window length	128	stochastic	2048	128	stochastic	2048
test window length	128	2048	2048	128	2048	2048
val_PPL ↓	2.551	**2.502**	2.523	2.291	**2.272**	2.283
HEplus/pass@10 ↑	12.20	12.80	15.24	24.39	24.39	26.83
ARC-c ↑	29.27	30.82	29.79	40.77	40.86	41.55
ARC-e ↑	65.75	68.71	67.15	75.05	74.46	74.80
Hellaswag ↑	45.37	45.51	45.91	53.34	53.59	53.69
MBPP/pass@10 ↑	24.20	30.60	28.80	44.60	47.20	45.40
NaturalQuestions ↑	13.51	12.47	13.19	22.21	23.45	23.04
PIQA ↑	73.83	74.43	73.99	77.26	77.97	76.55
RACE.high ↑	33.56	35.91	35.99	37.65	38.99	39.88
RACE.mid ↑	46.52	48.33	49.30	52.65	54.32	54.80
SIQA ↑	42.53	41.86	41.71	43.55	44.22	42.78
TriviaQA ↑	28.95	28.99	29.95	46.20	47.15	46.80
Winogrande ↑	62.04	59.27	59.51	67.88	67.64	65.75
average ↑	39.81	40.81	**40.88**	48.80	**49.52**	49.32

SWA layers here (Peng et al., 2023). On the other hand, windows of size 1024 and less show little degradation when reducing their train-time window size by half. Overall, models need to be trained at large windows sizes during training to be able to use those large windows during testing. However, we cannot allow the models to over-rely on the long softmax attention windows since those do not perform well on very long-context recall.

Stochastic window size To solve this dilemma, we introduce a training procedure that, throughout the training, stochastically alternates between a large window size and a small window size. Our hypothesis is that this will prevent the model from over-relying on the SWA layers for recall, since those do not perform well on very long-context tasks, while still making the model capable of using the larger window size at test time for better short-context reasoning. Moreover, to validate our experiments at a larger scale, we also train 7B parameter models using the same experimental setup as for the 1.4B models. For the 1.4B experiments, at each new batch of data, we set the window size to 128 with probability $p = 0.5$ or leave the default window size of 2048. At 7B scale, we use a slightly higher probability $p = 0.75$ of sampling the short window. We provide an ablation for the value of p in Appendix B. Finally, in order to make the model better use the larger test-time window of size 2048, we anneal the stochastic training procedure by not sampling the smaller window size anymore for the last 10% of training. We find that this short period of fixed window at the end of the training significantly helps short-context performance while not degrading long-context performance. We provide an ablation of the annealing in Appendix B.

Table 2 shows how training with a stochastic window size alternating between 128 and 2048 and annealing gives a short-context performance comparable to or even better than training with a fixed window size of 2048. In particular, at both the 1.4B scale and 7B scale, stochastic training gives considerably better short-context performance than having a fixed-sized window of 128. From a validation perplexity perspective, the stochastic window size actually outperforms all models trained with fixed window sizes all parameter scales. Therefore, training with a stochastic window size and testing with the longer window size at test time yields better results from a short-context task perspective compared to training with a using a short window size at both train and test time.

Compared to training with a long, fixed sliding window of size 2048, stochastic training gives comparable performance at the 1.4B scale and even slightly superior performance at the 7B scale on short-context reasoning tasks. This indicates that indeed, even if during training the model has seen the longer window size only part of the time, it is still able to take advantage of the longer window size for short-medium context reasoning tasks.

Long context performance of the stochastic training To ascertain whether or not this strategy will give a performance on long-context tasks as good as short-window variants, or not extrapolate

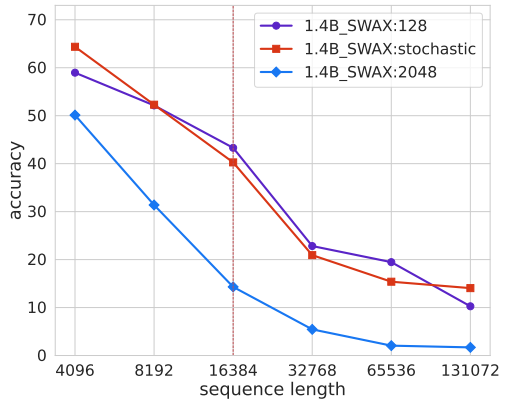

Figure 7: Average RULER NIAH accuracy of 1.4B SWAX models with different window sizes.

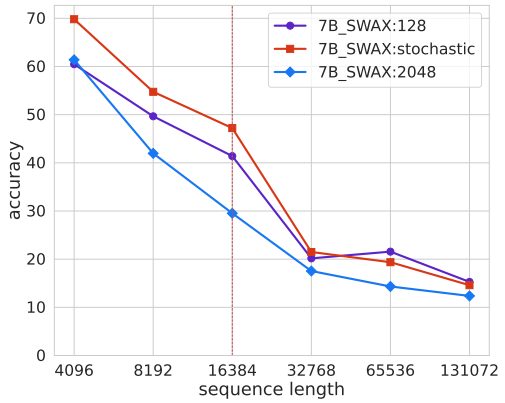

Figure 8: Average RULER NIAH accuracy of 7B SWAX models with different window sizes.

to longer sequence lengths like the long-window variants, we evaluated the performance of the stochastically trained SWAX models with annealing on the last 10% of the training.

Figures 7 and 8 show that on RULER's long-context recall tasks, the models trained with a stochastic window size and annealing perform on par or are better than the model trained with the short window of 128, and drastically better than the hybrid model trained with a window of 2048 tokens. For instance, at 1.4B parameter scale depicted in Figure 7, the stochastic training SWAX model performs similarly to the short-window SWAX. Figure 8 shows that stochastic training also improves the long-context performance of models at the 7B scale. At 7B scale, compared to using a fixed, long sliding window of size 2048, the stochastic training gives much better retrieval accuracy at all sequence lengths. Furthermore, just as at 1.4B scale, stochastic training gives similar or even superior long-context performance compared to using a short window throughout training. Overall, these results show that a stochastic window size at training time maintains all the benefits of having a short window for long-context recall, and most if not all of the benefits of having a longer window for short/medium-context reasoning tasks.

Moreover, this result further confirms that the poor performance of hybrids with a long sliding window is not intrinsically due to using a long sliding window at test time. Instead, these results show that the poor length-generalization of hybrids with long sliding windows is due to the training procedure. Indeed, if the model is allowed to use the long sliding window at all times throughout training, it will over-rely on the more precise softmax attention of the sliding window even for recall tasks, which will not extrapolate to longer sequence lengths, and will under-utilize the Linear Attention Layers for the long-term recall task. On the contrary, if during training the model is not allowed to over-rely on the long sliding window, and is instead stochastically forced to use a shorter window, then the linear attention will have to be trained on the medium/long-term recall task and the model will generalize to longer sequence lengths at test time. Since, essentially, this amounts to stochastically reducing the capacity of the model in order to make it more robust, this stochastic training can be seen as a form of dropout (Srivastava et al., 2014) on the attention mechanism.

5 Conclusion

Through an empirical analysis of hybrid architectures, we evidence the counter-intuitive fact that shorter sliding windows lead to better length-extrapolation on retrieval tasks. Moreover, we introduce a training procedure which stochastically changes the window size throughout training to prevent over-reliance on the long softmax attention windows. This training procedure allows the model to profit from both the strong performance of longer sliding windows offer on short context tasks, and the length-extrapolation ability of Linear Attention layers enabled by shorter windows.

REFERENCES

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, Dylan Zinsley, James Zou, Atri Rudra, and Christopher Ré. Simple linear attention language models balance the recall-throughput tradeoff, 2025. URL https://arxiv.org/abs/2402.18668.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021. URL https://arxiv.org/abs/2108.07732.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. URL https://arxiv.org/abs/1607.06450.

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.

Maximilian Beck, Korbinian Pöppel, Phillip Lippe, and Sepp Hochreiter. Tiled Flash Linear Attention: More efficient linear rnn and xlstm kernels. *arXiv*, 2503.14376, 2025a. URL https://arxiv.org/abs/2503.14376.

Maximilian Beck, Korbinian Pöppel, Phillip Lippe, Richard Kurle, Patrick M. Blies, Günter Klambauer, Sebastian Böck, and Sepp Hochreiter. xlstm 7b: A recurrent llm for fast and efficient inference, 2025b. URL https://arxiv.org/abs/2503.13427.

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

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014. URL https://arxiv.org/abs/1412.3555.

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.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Xin Dong, Yonggan Fu, Shizhe Diao, Wonmin Byeon, Zijia Chen, Ameya Sunil Mahabaleshwarkar, Shih-Yang Liu, Matthijs Van Keirsbilck, Min-Hung Chen, Yoshi Suhara, Yingyan Lin, Jan Kautz, and Pavlo Molchanov. Hymba: A hybrid-head architecture for small language models, 2024. URL https://arxiv.org/abs/2411.13676.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning, 2017. URL https://arxiv.org/abs/1702.03118.

Lizhe Fang, Yifei Wang, Zhaoyang Liu, Chenheng Zhang, Stefanie Jegelka, Jinyang Gao, Bolin Ding, and Yisen Wang. What is wrong with perplexity for long-context language modeling?, 2024. URL https://arxiv.org/abs/2410.23771.

Daniel Y. Fu, Tri Dao, Khaled K. Saab, Armin W. Thomas, Atri Rudra, and Christopher Ré. Hungry hungry hippos: Towards language modeling with state space models, 2023. URL https://arxiv.org/abs/2212.14052.

Jonas Gehring, Kunhao Zheng, Jade Copet, Vegard Mella, Quentin Carbonneaux, Taco Cohen, and Gabriel Synnaeve. Rlef: Grounding code llms in execution feedback with reinforcement learning, 2025. URL https://arxiv.org/abs/2410.02089.

Google DeepMind Gemma Team. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2024. URL https://arxiv.org/abs/2312.00752.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 11 1997. doi: 10.1162/neco.1997.9.8.1735.

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?, 2024. URL https://arxiv.org/abs/2404.06654.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint*, 2024.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention, 2020. URL https://arxiv.org/abs/2006.16236.

Bo Liu, Rui Wang, Lemeng Wu, Yihao Feng, Peter Stone, and Qiang Liu. Longhorn: State space models are amortized online learners, 2024. URL https://arxiv.org/abs/2407.14207.

Hanxiao Liu, Zihang Dai, David R. So, and Quoc V. Le. Pay attention to mlps, 2021. URL https://arxiv.org/abs/2105.08050.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=1qvx610Cu7.

Team NVIDIA. Nemotron-h: A family of accurate and efficient hybrid mamba-transformer models, 2025. URL https://arxiv.org/abs/2504.03624.

OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025. URL https://arxiv.org/abs/2508.10925.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. Yarn: Efficient context window extension of large language models, 2023. URL https://arxiv.org/abs/2309.00071.

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, 2025. URL https://arxiv.org/abs/2406.07522.

Ricardo Buitrago Ruiz and Albert Gu. Understanding and improving length generalization in recurrent models, 2025. URL https://arxiv.org/abs/2507.02782.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023. URL https://arxiv.org/abs/2104.09864.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): Rnns with expressive hidden states, 2025. URL https://arxiv.org/abs/2407.04620.

Dustin Wang, Rui-Jie Zhu, Steven Abreu, Yong Shan, Taylor Kergan, Yuqi Pan, Yuhong Chou, Zheng Li, Ge Zhang, Wenhao Huang, and Jason Eshraghian. A systematic analysis of hybrid linear attention, 2025. URL https://arxiv.org/abs/2507.06457.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

Guangxuan Xiao. Why stacking sliding windows can't see very far. https://guangxuanx.com/blog/stacking-swa.html, 2025.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training, 2024. URL https://arxiv.org/abs/2312.06635.

Biao Zhang and Rico Sennrich. Root mean square layer normalization, 2019. URL https://arxiv.org/abs/1910.07467.

Jianyu Zhang and Léon Bottou. Memory mosaics at scale. *arXiv preprint arXiv:2507.03285*, 2025.

Shu Zhong, Mingyu Xu, Tenglong Ao, and Guang Shi. Understanding transformer from the perspective of associative memory, 2025. URL https://arxiv.org/abs/2505.19488.

SUPPLEMENTARY MATERIAL

A RESULTS OF PURE SWA MODELS

In section 4.3 we hypothesize that the worse performance of SWAX models with long windows comes from the model utilizing the SWA layers instead of the xLSTM layers. To further confirm this hypothesis, we train a 1.4B pure SWA model with a window size of 2048 and compare its performance to the SWAX model with the same window size. If the hypothesis that the SWAX model relies on the SWA layers for recall is valid, then we expect its performance to be similar to that of a pure SWA architecture.

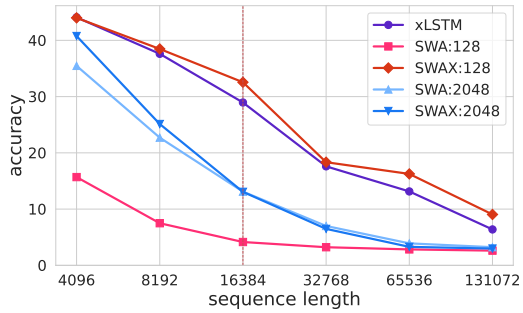

Figure 9: Average accuracy of 1.4B parameter models over all RULER NIAH tasks.

Figure 9 shows that indeed, the SWAX model trained with a window of 2048 performs very similarly to the pure SWA architecture. On the other hand, the accuracy the SWAX model trained with a window of 128 is very dissimilar from that of the pure SWA model with a window of 128. This further evidences the fact that low long-context performance of hybrid models with long windows comes from the model over-relying on the SWA layers for long-context recall instead of using the xLSTM layers.

B ABLATION ON STOCHASTIC SAMPLING PROBABILITY AND ANNEALING OF STOCHASTICITY

In this experiment we perform an ablation on the stochastic sampling probability p its schedule during training for the two model sizes 1.4B and 7B we consider in our study. Table 3 shows that, at 7B scale, a higher probability of sampling the small window during training is necessary to significantly improve short and long context performance compared to the probability of 0.5 which worked at 1.4B scale. Looking at the impact of annealing, i.e. using a stochastic window size for the first X% of training, at both 1.4B and 7B scale, annealing improves short-context performance compared to keeping the stochasticity until the end of training. At 7B scale, the annealed SWAX model even performs better on short-context than the SWAX model trained with a fixed window size of 2048. In terms of long-context performance, compared to keeping the stochasticity until the end of training, annealing slightly degrades the long-context performance at 1.4B scale but keeps or even slightly improves long-context performance at 7B. We believe that exploring different annealing procedures might provide even better short-context performance improvements while — at the same time — keeping good long-context performance.

model parameters	xLSTM 7B	SWAX 7B						SWAX 1.4B	
train-time window test-time window	NA NA	128 128	p=0.9 2048	p=0.75 2048	$p^{90\%}$=0.75 2048	p=0.5 2048	2048 2048	p=0.5 2048	$p^{90\%}$=0.5 2048
niah_single	61.20	62.43	58.99	**63.36**	63.20	55.27	53.46	62.61	61.59
niah_multiquery	**44.18**	34.78	35.95	32.72	32.23	17.94	21.62	30.85	27.42
niah_multikey	10.14	9.96	13.32	17.23	**17.86**	14.34	12.29	14.52	12.19
niah_multivalue	**39.28**	26.11	23.55	27.32	27.56	19.48	17.30	30.63	27.63
niah_average	37.19	34.76	34.55	37.73	**37.87**	30.78	29.52	**36.61**	34.55
HEplus/pass@10	25.00	24.39	25.61	23.17	24.39	21.95	**26.83**	13.41	12.80
arc-c	37.42	40.77	40.00	40.86	40.86	40.09	**41.55**	32.45	30.82
arc-e	73.32	**75.05**	74.76	74.50	74.46	74.63	74.80	67.23	28.71
hella	52.95	53.34	53.34	53.48	53.59	53.47	**53.69**	45.61	45.51
mbpp/pass@10	43.80	44.60	43.60	42.80	**47.20**	45.80	45.40	28.00	30.60
nq	22.12	22.21	23.12	23.16	**23.45**	22.58	23.04	12.95	12.47
piqa	76.93	77.26	76.71	**78.07**	77.97	77.31	76.55	74.32	74.43
race.high	37.02	37.65	37.71	38.74	38.99	38.74	**39.88**	34.82	35.91
race.mid	52.85	52.65	54.11	54.32	54.32	53.90	**54.80**	48.26	48.33
siqa	43.96	43.55	44.06	44.01	44.22	**44.37**	42.78	41.91	41.86
tqa	46.39	46.20	**47.44**	46.90	47.15	46.55	46.80	29.35	28.99
wino	66.06	67.88	**68.51**	67.32	67.64	66.77	65.75	59.20	59.27
short-average	48.15	48.80	49.08	48.95	**49.52**	48.85	49.32	40.62	**40.81**

Table 3: NIAH and downstream tasks accuracy for 7B models. p indicates the probability of using a window of 128 for a batch, otherwise using a window of 2048. $p^{90\%}$ indicates annealing, i.e., only doing the stochastic window size for the first 90% of the training and then using a fixed window size of 2048 for the rest of training. NIAH single and multikey results are the average overall all 3 sub-tasks for each.