Samba: Simple Hybrid State Space Models for Efficient Unlimited Context Language Modeling

Liliang Ren, Yang Liu, Yadong Lu, Yelong Shen, Chen Liang, Weizhu Chen

Microsoft
University of Illinois at Urbana-Champaign

References

Samba

- Github : https://github.com/microsoft/Samba
- Hugging Face : https://huggingface.co/papers/2406.07522
- Paper : https://arxiv.org/html/2406.07522v1
  https://arxiv.org/abs/2406.07522

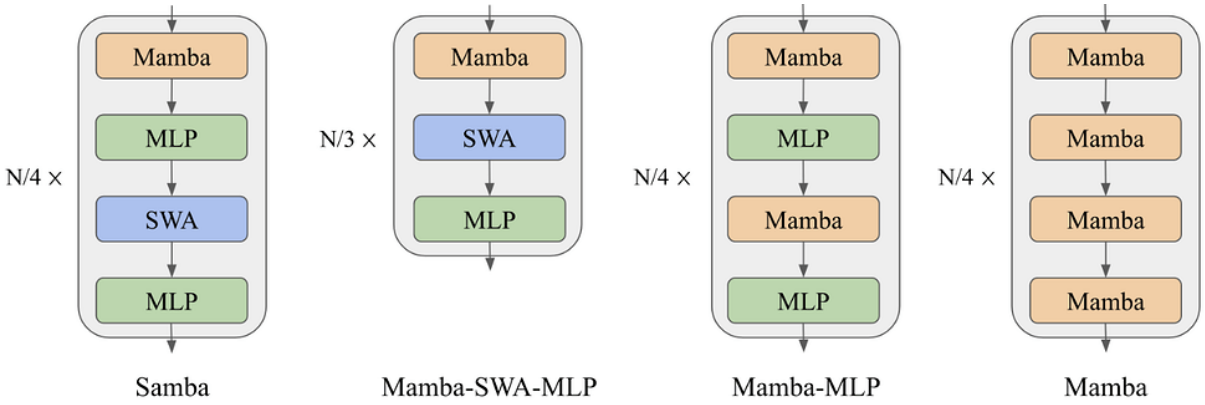
Introduction

➢ **Outlines**

- Samba is a simple powerful **hybrid model** with **an unlimited context length**. Its architecture is frustratingly simple:

- **Samba = Mamba + MLP + SWA(Sliding Window Attention) + MLP stacking** at the layer level.

- Our largest model, **Samba-3.8B**, is trained on **3.2 Trillion tokens** from the **Phi3 dataset**, outperforming **Phi3-mini** on major benchmarks (e.g. MMLU, GSM8K and HumanEval).
- Samba can also **achieve perfect long-context retrieval ability with minimal instruction tuning**, while still **maintaining its linear complexity with respect to sequence length**.
- This ability leads to the impressive performance of Samba-3.8B-instruct on downstream tasks such as **long-context summarization**.



Introduction

➢ Abstract

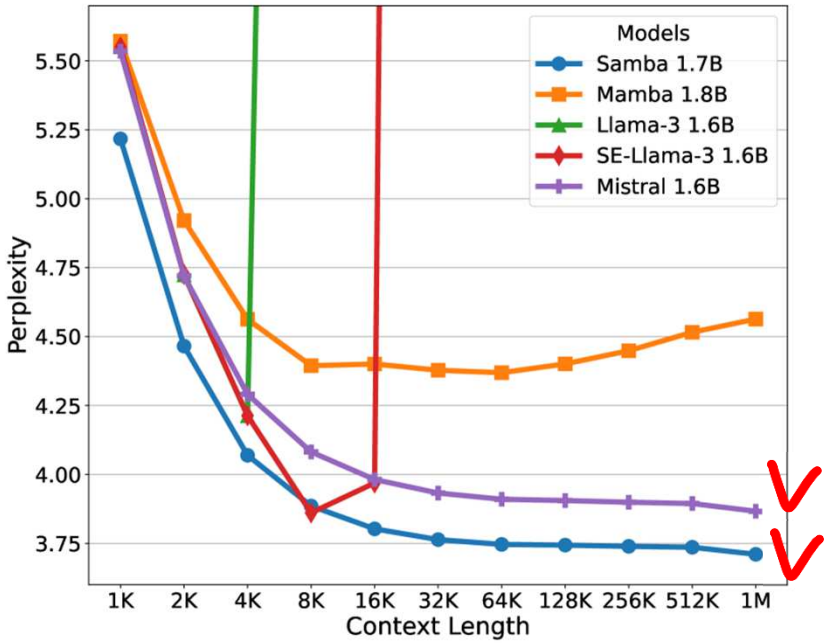
- **Efficiently modeling sequences with infinite context length** has been a **long-standing problem**. Past works **suffer from** either **the quadratic computation complexity** or **the limited extrapolation ability on length generalization**.

➢ **Samba**, **a simple hybrid architecture** that **layer-wise** combines **Mamba, a selective State Space Model (SSM)**, with **Sliding Window Attention (SWA)**.

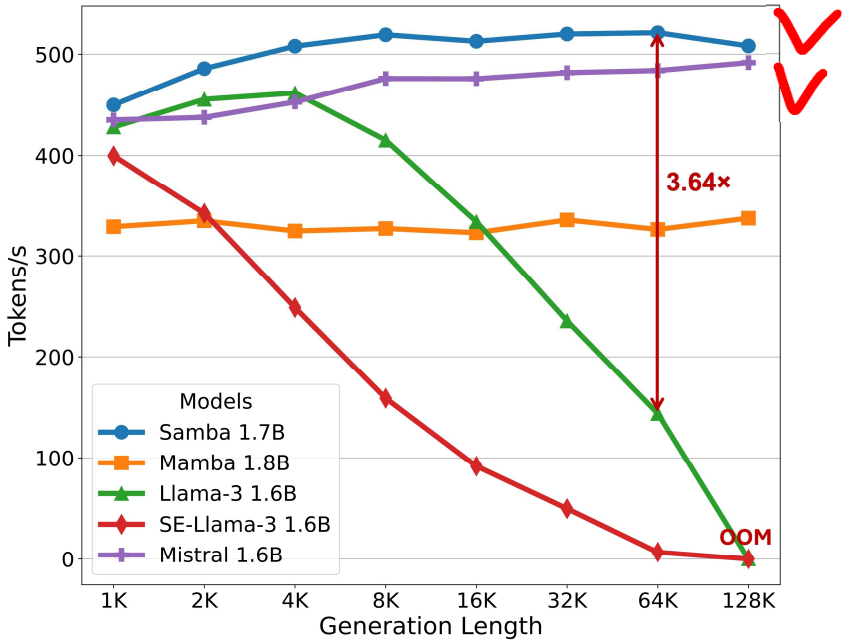
- **Selectively compresses a given sequence into recurrent hidden states**
- While still **maintaining the ability to precisely recall memories with the *attention mechanism***.

- **Samba-3.8B** with **3.2T training tokens** : Outperforms the state-of-the-art models based on pure attention or SSMs on a wide range of benchmarks.
- When trained on *4K length sequences*, Samba can be efficiently extrapolated to *256K context length* with perfect memory recall and show improved token predictions up to *1M context length*.
- As a linear-time sequence model, Samba enjoys a **3.73× higher throughput** compared to **Transformers** with ***grouped-query attention*** when processing user prompts of 128K length, and **3.64× speedup** when generating 64K tokens with unlimited streaming.

Introduction



(a) **Perplexity** on the test set of **Proof-Pile**



(b) Decoding throughput with a batch size of 16

Figure 1: Samba shows improved prediction up to 1M tokens in the Proof-Pile test set while achieving a **3.64× faster decoding** throughput than **the Llama-3 architecture** [Met24] (a state-of-the-art ***Transformer*** [VSP+17] with ***Grouped-Query Attention*** [ALTdJ+23]) on 64K generation length. We also include an ***SE-Llama-3 1.6B baseline*** which applies the ***SelfExtend*** [JHY+24] approach for *zero-shot length extrapolation*. Throughput measured on **a single A100 80GB GPU**. All models are trained on the Phi-2 [LBE+23] dataset with 4K sequence length.

Introduction

➢ Related Works

- **Attention-based models** [VSP+17, BCB14]
  - ✓ Dominated the neural architectures of LLMs [RWC+19, BMR+20, Ope23, BCE+23]
  - ✓ Ability to capture ***complex long-term dependencies*** and the efficient ***parallelization*** for large-scale training [DFE+22].

- **State Space Models (SSMs)** [GGR21, SWL23, GGGR22, GD23]
  - ✓ Emerged as a promising alternative
  - ✓ ***Linear computation complexity*** and the potential for better ***extrapolation*** to longer sequences than seen during training.
  - ❖ **Mamba**[GD23], a variant of SSMs equipped with **selective state spaces**; Strong empirical performance and efficient hardware-aware implementation.

- Recent work shows
  - ✓ **Transformers** have ***poorer modeling capacities*** than input-dependent SSMs in state tracking problems [MPS24].
  - ✓ However, **SSMs struggle with *memory recall*** due to their **Markovian nature** [AET+23], and experimental results on information retrieval-related tasks [FDS+23, WDL24, AEZ+24], have shown that SSMs are not as competitive as their attention-based counterparts.

- Previous works (Zuo et al., 2022; Fu et al., 2023; Ma et al., 2023; Ren et al., 2023) have explored **various approaches to hybridize SSMs with the attention mechanism**, but none have demonstrated significantly better language modeling performance compared to SOTA Transformer architectures.

- **Existing length *extrapolation* techniques** (Han et al., 2023; Xiao et al., 2023; Jin et al., 2024) designed for attention mechanisms are **constrained by quadratic computational complexity** or **insufficient context extrapolation performance**, particularly when evaluated under perplexity metrics.

Introduction

➢ Summary SAMBA

- We introduce **SAMBA**, a simple neural architecture that harmonizes the strengths of both **the SSM and the attention-based models**, while **achieving a potentially infinite length extrapolation with linear time complexity**.
- SAMBA combines **SSMs** with **attention** through **layer-wise interleaving Mamba** (Gu & Dao, 2023), **SwiGLU** (Shazeer, 2020), and **Sliding Window Attention (SWA)** (Beltagy et al., 2020).
  - ✓ ***Mamba*** layers capture the time-dependent semantics and provide *a backbone for efficient decoding*,
  - ✓ ***SWA*** *fills in the gap modeling complex, non-recurrent dependencies*.
- We **scale** SAMBA with 421M, 1.3B, 1.7B and **up to 3.8B parameters with 3.2T tokens**.
- The largest **3.8B post-trained model** achieves a **47.9 score for MMLU-Pro** (Hendrycks et al., 2021), **70.1 for HumanEval** (Chen et al., 2021), and **86.4 for GSM8K** (Cobbe et al., 2021), substantially outperforming strong open source language models up to 8B parameters, as detailed in Table 8.
- Despite being **pre-trained in the 4K sequence length**, SAMBA can be **extrapolated to 1M length in zero shot with improved perplexity on Proof-Pile** (Zhangir Azerbayev & Piotrowski, 2022), achieving a **256× extrapolation ratio**, while still **maintaining the linear decoding time complexity with unlimited token streaming**, as shown in Figure 2.
  - ✓ When instruction-tuned in a 4K context length with only 500 steps, SAMBA can be extrapolated to a 256K context length with perfect memory recall in Passkey Retrieval (Mohtashami & Jaggi, 2023).
  - ✓ In contrast, the fine-tuned SWA-based model simply cannot recall memories beyond 4K length.

Introduction

➢ Summary SAMBA

- **Samba** is **the first hybrid model** showing that
- ***Linear complexity models*** can be substantially better than SOTA *Transformer models* on **short-context tasks at large scale**,
- While still being able to **extrapolate to extremely long sequences** under the perplexity metric.

➢ Methodology

- Explore **different hybridization strategies** consisting of the layers of **Mamba**, **Sliding Window Attention (SWA),** and **Multi-Layer Perceptron** (Shazeer, 2020; Dauphin et al., 2016).
- Conceptualize the functionality of **Mamba as the capture of recurrent sequence structures**, **SWA as the precise retrieval of memory**, and **MLP as the recall of factual knowledge**.
- Also explore **other linear recurrent layers** including **Multi-Scale Retention** (Sun et al., 2023) and **GLA** (Yang et al., 2023) as **potential substitutions for Mamba** in Section 3.2.

❖ **Our goal of hybridization** is to **harmonize between these distinct functioning blocks** and **find an efficient architecture for language modeling with unlimited length extrapolation ability**

METHODOLOGY

2.1. ARCHITECTURE

- Three kinds of layerwise hybridization strategies on the 1.7B scale: **Samba**, **Mamba-SWA-MLP**, and **Mamba-MLP**.
- Other hybridization approaches with **full self-attention** on smaller scales
- **Number of layers N** is set to **48** for **Samba, Mamba-MLP, and Mamba**, while **Mamba-SWA-MLP has 54 layers**, so each model has approximately **1.7B** parameters.

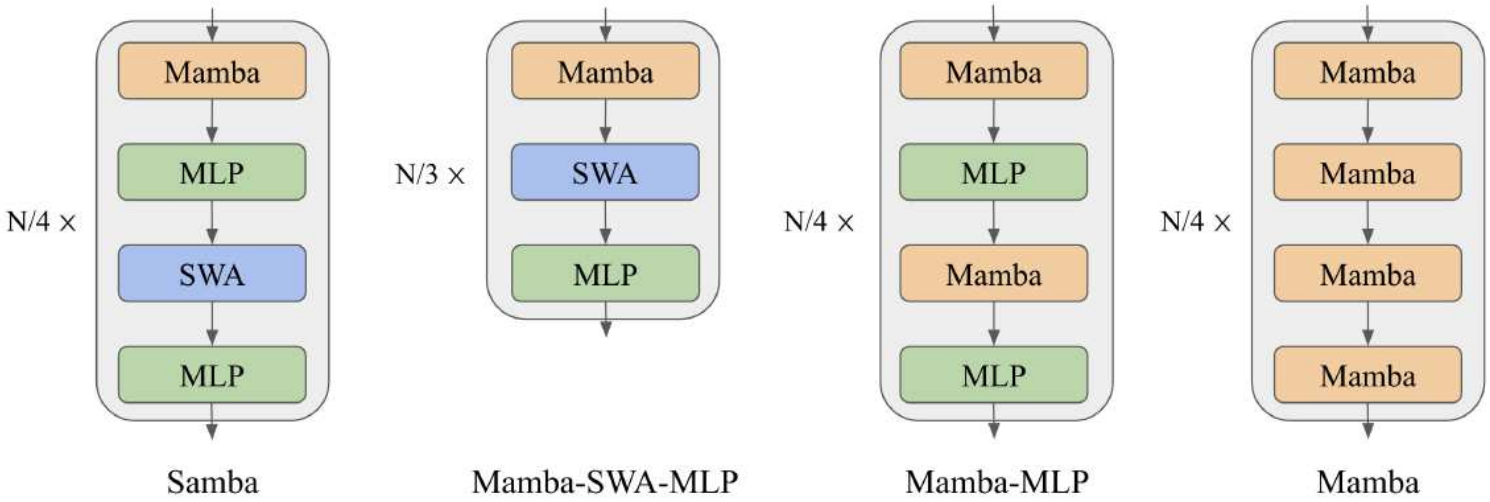


Figure 1: From left to right: Samba, Mamba-SWA-MLP, Mamba-MLP, and Mamba.

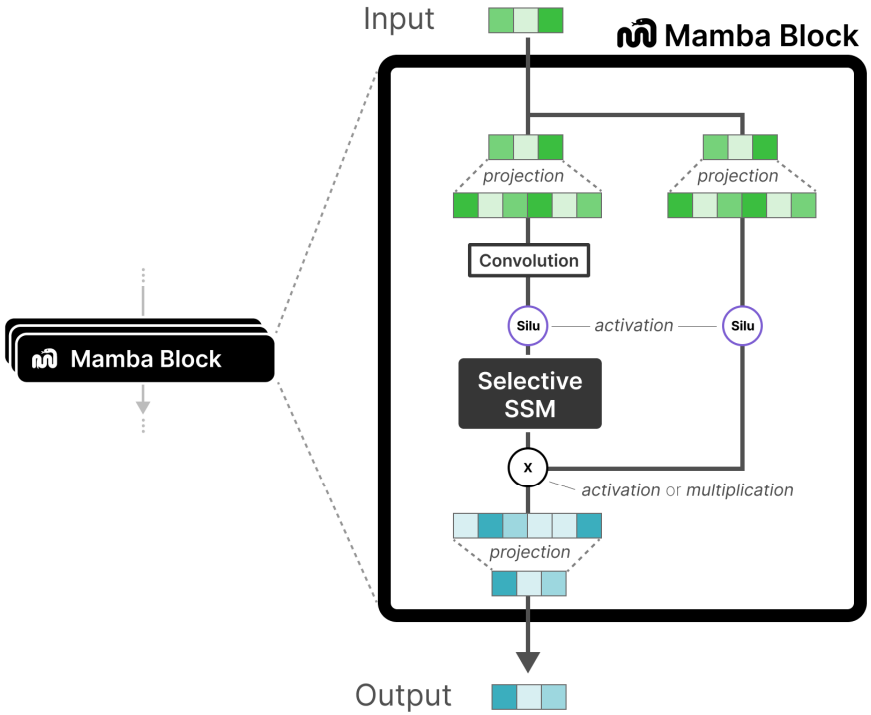
The illustrations depict the layer-wise integration of Mamba with various configurations of Multi-Layer Perceptrons (MLPs) and Sliding Window Attention (SWA).

We assume the total number of intermediate layers to be N, and omit the embedding layers and output projections for simplicity. Pre-Norm (Xiong et al., 2020; Zhang & Sennrich, 2019) and skip connections (He et al., 2016) are applied for each of the intermediate layers.

METHODOLOGY

2.1.1 MAMBA LAYER

- Mamba (Gu & Dao, 2023) : SSM-based model with selective state spaces
- Enables ***input-dependent gating*** to both *the* ***recurrent states*** and *the* ***input representation*** for a soft selection of the input sequence elements.
- Given an input sequence representation $\mathbf{X} \in \mathbb{R}^{n \times d_m}$, where n is the length of the sequence and d_m is the hidden size,



1) Mamba first **expands the inputs** to a higher dimension de, i.e.,

$$\mathbf{H} = \mathbf{X}\mathbf{W}_{\text{in}} \in \mathbb{R}^{n \times d_e}$$

✓ $\mathbf{W}_{in} \in \mathbb{R}^{d_m \times d_e}$ is a learnable projection matrix

2) Then a **Short Convolution (SC)** (Poli et al., 2023) operator is applied to smooth the input signal,

$$\mathbf{U} = \text{SC}(\mathbf{H}) = \text{SiLU}(\text{DepthwiseConv}(\mathbf{H}, \mathbf{W}_{\text{conv}})) \in \mathbb{R}^{n \times d_e} \quad (1)$$

✓ $\mathbf{W}_{conv} \in \mathbb{R}^{k \times d_e}$ and the kernel size k is set to 4 for hardware-aware efficiency

✓ The *Depthwise* Convolution (He et al., 2019) is applied over the sequence dimension followed by a SiLU (Elfwing et al., 2017) activation function.

2.1.1 MAMBA LAYER

3) Then **the selective gate** is calculated through *a low-rank projection* followed by *Softplus* (Zheng et al., 2015),

$$\Delta = \text{Softplus}(\mathbf{U}\mathbf{W}_\text{r}\mathbf{W}_\text{q} + \mathbf{b}) \in \mathbb{R}^{n \times d_e} \quad (2)$$

- ✓ $\mathbf{W}_r \in \mathbb{R}^{d_e \times d_r}$, $\mathbf{W}_q \in \mathbb{R}^{d_r \times d_e}$ and d_r is the low-rank dimension.
- ✓ $\mathbf{b} \in \mathbb{R}^{d_e}$ is carefully initialized so that $\Delta \in [\Delta_{min}, \Delta_{max}]$ after the initialization stage.
- ✓ $[\Delta_{min}, \Delta_{max}]$ = [0.001, 0.1], find that these values are not sensitive to language modeling performance under the perplexity metric.

- The **input dependence** is introduced for the parameters **B** and **C** of SSM

$$\mathbf{B} = \mathbf{U}\mathbf{W}_\text{b} \in \mathbb{R}^{n \times d_s}$$
$$\mathbf{C} = \mathbf{U}\mathbf{W}_\text{c} \in \mathbb{R}^{n \times d_s}$$

- ✓ d_s is the state dimension

4) For each time step $1 \le t \le n$, the **recurrent inference of the Selective SSM (S6)** is performed in an expanded state space $\mathbf{Z}_t \in \mathbb{R}^{d_e \times d_s}$

$$\mathbf{Z}_t = \exp(-\Delta_t \odot \exp(\mathbf{A})) \odot \mathbf{Z}_{t-1} + \Delta_t \odot (\mathbf{B}_t \otimes \mathbf{U}_t) \in \mathbb{R}^{d_e \times d_s}$$
$$\mathbf{Y}_t = \mathbf{Z}_t\mathbf{C}_t + \mathbf{D} \odot \mathbf{U}_t \in \mathbb{R}^{d_e}$$

- ✓ $\mathbf{Z}_0 = \mathbf{0}$
- ✓ $\odot$ means the point-wise product, $\otimes$ means the outer product and exp means the point-wise natural exponential function
- ✓ $\mathbf{D} \in \mathbb{R}^{d_e}$ is a learnable vector initialized as $\mathbf{D}_i = 1$
- ✓ $\mathbf{A} \in \mathbb{R}^{d_e \times d_s}$ is a learnable matrix initialized as $A_{ij} = \log(j), 1 \le j \le d_s$, following the S4DReal (Gu et al., 2022) initialization

2.1.1 MAMBA LAYER

- In practice, Mamba implements a **hardware-aware parallel scan algorithm** for **efficient parallelizable training**.

5) **The final output** is obtained through **a gating mechanism** similar to Gated Linear Unit (Shazeer, 2020; Dauphin et al., 2016),

$$\mathbf{O} = \mathbf{Y} \odot \mathrm{SiLU}(\mathbf{X}\mathbf{W}_{\mathrm{g}})\mathbf{W}_{\mathrm{out}} \in \mathbb{R}^{n \times d_m}$$

- ✓ $\mathbf{W}_g \in \mathbb{R}^{d_m \times d_e}$, and $\mathbf{W}_{out} \in \mathbb{R}^{d_e \times d_m}$ are learnable parameters.
- ✓ We set $d_e = 2d_m$, $d_r = d_m/16$, $d_s = 16$

- ❖ The **Mamba layer in SAMBA** is expected to **capture the time-dependent semantics of the input sequence through its recurrent structure**.
- ❖ The **input selection mechanism** in the Mamba layer enables the model to **focus on relevant inputs**, thereby allowing the model to **memorize important information in the long term.**

METHODOLOGY

2.1.2 SLIDING WINDOW ATTENTION (SWA) LAYER

- **Sliding Window Attention** (Beltagy et al., 2020) layers to address the limitations of Mamba layers **in capturing non-recurrent dependencies in sequences**.

- Our **SWA layer** operates on a window size **w = 2048** that slides over the input sequence (*the computational complexity remains linear with respect to the sequence length*).

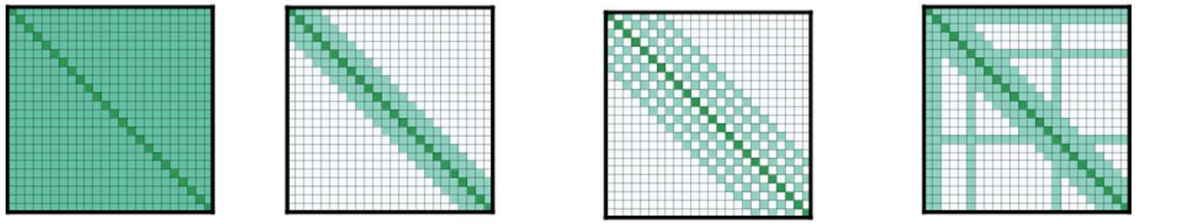
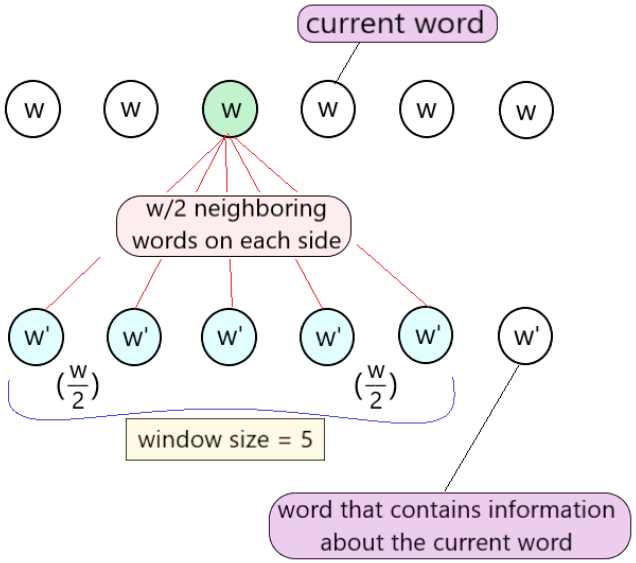
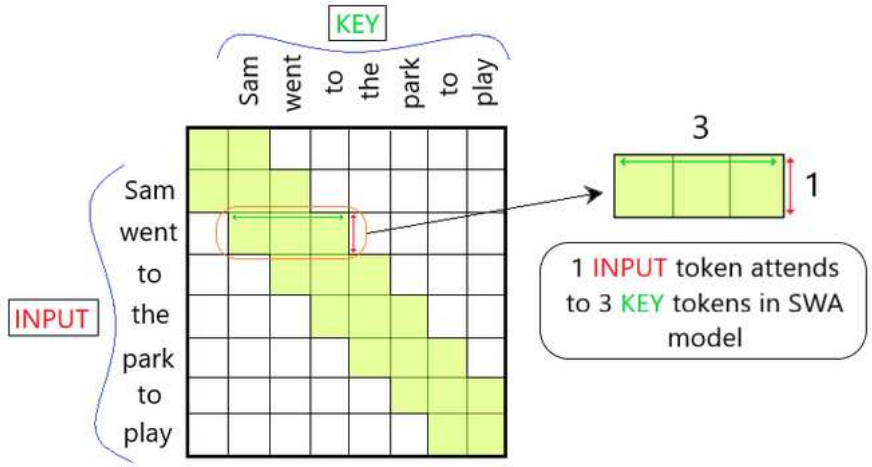


Figure 2: Comparing the full self-attention pattern and the configuration of attention patterns in our Longformer



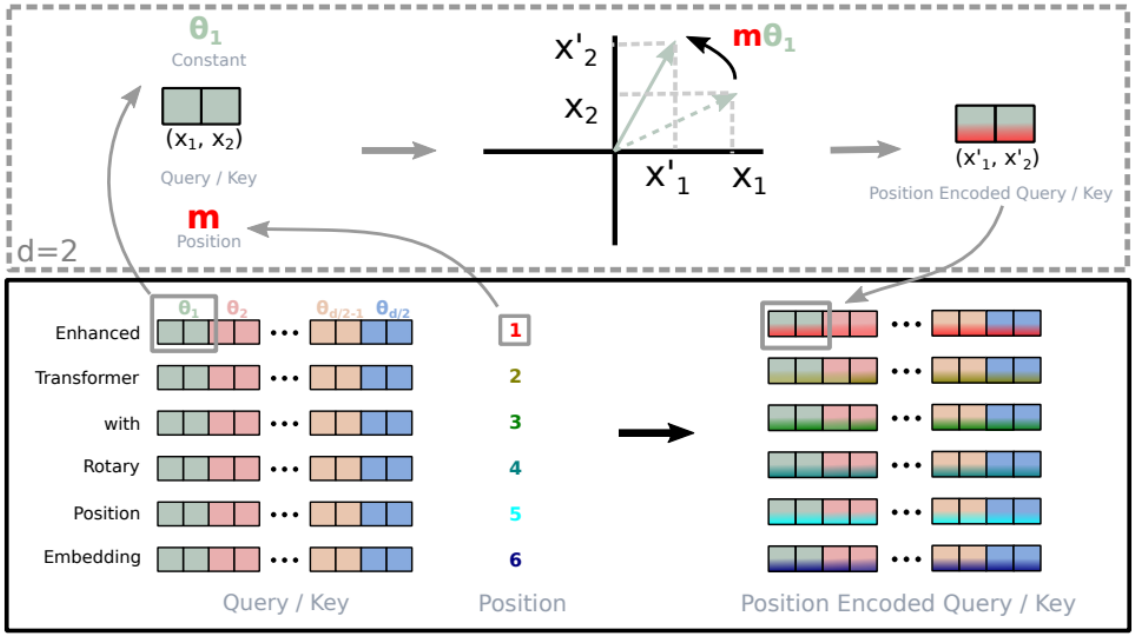
Working of sliding window attention model



☞https://www.geeksforgeeks.org/sliding-window-attention/ Intuition Example

2.1.2 SLIDING WINDOW ATTENTION (SWA) LAYER

- **RoPE** (Su et al., 2021) is applied within the sliding window, with a **base frequency of 10,000**.
- **By directly accessing the contents in the context window through attention,** the SWA layer can **retrieve high-definition signals from the middle to short-term history** that cannot be clearly captured by the recurrent states of Mamba.



☞ RoFormer: Enhanced Transformer with Rotary Position Embedding

https://arxiv.org/abs/2104.09864

https://kimjy99.github.io/논문리뷰/roformer/

- RoPE는 회전 행렬로 절대적 위치를 인코딩하는 한편, self-attention 공식에 명시적인 상대적 위치 종속성을 통합한다.
- RoPE는 시퀀스 길이 유연성을 가지고 있으며, 상대적 거리가 증가함에 따라 토큰 간 종속성이 감소하며, linear self-attention에 상대적 위치 인코딩을 장착할 수 있다.
- RoPE를 갖춘 향상된 Transformer인 RoFormer는 다양한 장문 텍스트 분류 벤치마크 데이터셋에서 baseline에 비해 더 나은 성능을 보인다.

Implementation of RoPE (RoFormer: Enhanced Transformer with **Rotary Position Embedding**)

- Word Embedding 벡터를 Complex(복소수) 형태로 변환시킨 후 이를 rotation함. 그렇게 되면 Word Embedding의 절대적인 값이 변경됨
- Self attention에서 사용하는 '내적' 은 내적 하는 벡터 사이 각도에 대한 함수이므로 두 벡터에 대한 상대 거리가 보존될 수 있다. → 그래서 오히려 (이전에 사용해온) 절대 Position 정보를 더해주는 방식은 효율적이지 못하다는 것.

2.1.2 SLIDING WINDOW ATTENTION (SWA) LAYER

- Use **FlashAttention 2** (Dao, 2023) for the efficient implementation of self-attention throughout this work.
- Choose the **2048 sliding window size** for efficiency consideration; FlashAttention 2 has the same training speed as Mamba's selective parallel scan at the sequence length of 2048 based on the measurements in (Gu & Dao, 2023).

❖ **FlashAttention-2 (2023)** [☞ https://news.hada.io/topic?id=9892]

- GPT-4(32k), MPT(65k), Claude(100k) 등 더 긴 컨텍스트를 가진 언어모델이 출현
- 트랜스포머의 컨텍스트 길이를 확장하는 것은 런타임&메모리 요구사항이 4제곱으로 증가하기 때문에 어려움
- FlashAttention(2022)은 메모리 사용량을 줄이고 어텐션 속도를 증가시켜서 다양한 곳에서 이용됨
- FlashAttention-2는 이전 버전보다 2배 빠르고, A100 GPU에서 최대 230 TFLOP/s 의 성능을 제공
- GPT 형태의 언어모델 훈련에서는 최대 225 TFLOPS까지 도달했음 (72% 모델 FLOP 활용도)
- 알고리즘을 조정하여 non-matmul FLOPs를 줄였음
- 더 나은 병렬화, 각 스레드 블록에서의 작업 분할방법 변경
- Head Dimensions 개수를 128에서 256개로 확장

➢ Tri Dao, Stanford Univ., [☞https://tridao.me/blog/]

- [2022] FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness
- [2023] FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning
- [2024] FlashAttention-3: Fast and Accurate Attention with Asynchrony and Low-precision

2.1.2 SLIDING WINDOW ATTENTION (SWA) LAYER

❖ **FlashAttention 1**

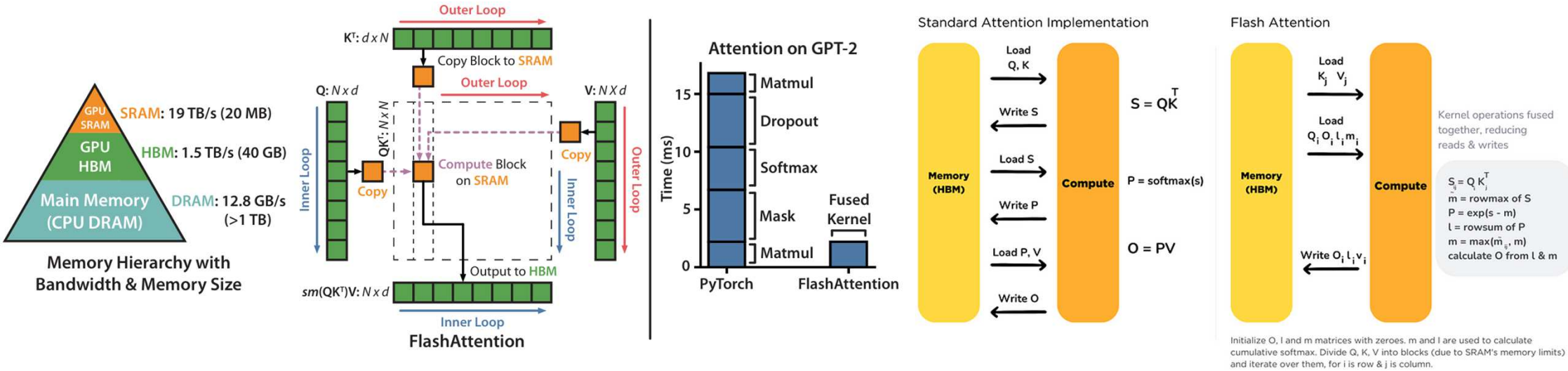
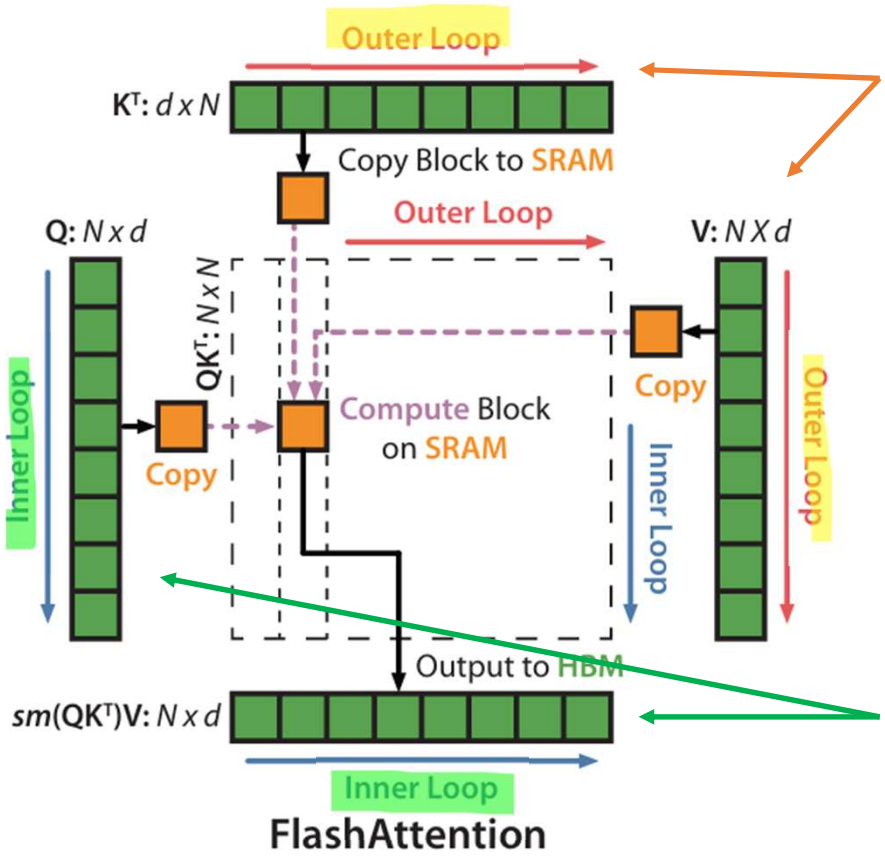


Figure 1: **Left**: FlashAttention uses *tiling* to prevent materialization of the large NxN attention matrix (dotted box) on (relatively) slow GPU HBM. In the outer loop (red arrows), FlashAttention loops through blocks of the K and V matrices and loads them to fast on-chip SRAM. In each block, FlashAttention loops over blocks of Q matrix (blue arrows), loading them to SRAM, and writing the output of the attention computation back to HBM.

Right: Speedup over the PyTorch implementation of attention on GPT-2. FlashAttention does not read and write the large NxN attention matrix to HBM, resulting in a **7.6x** speedup on the attention computation

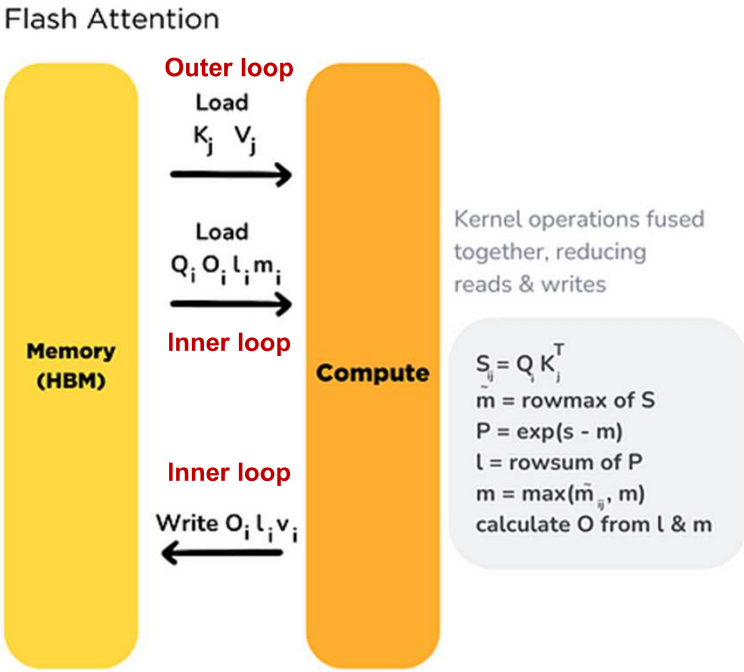
2.1.2 SLIDING WINDOW ATTENTION (SWA) LAYER

❖ **FlashAttention 1**



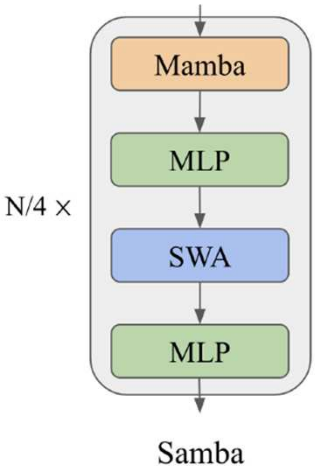
In the outer loop (red arrows), FlashAttention loops through blocks of the K and V matrices and loads them to fast on-chip SRAM.

In each block, FlashAttention loops over blocks of Q matrix (blue arrows), loading them to SRAM, and writing the output of the attention computation back to HBM.



2.1.3 MULTI-LAYER PERCEPTRON (MLP) LAYER

- The **MLP layers** in SAMBA serve as the architecture's primary mechanism for **nonlinear transformation** and **recall of factual knowledge** (Dai et al., 2022).
- We use **SwiGLU** (Shazeer, 2020) for all the models trained in this paper and denote its intermediate hidden size as d_p.
- As shown in Figure 1, Samba applies **separate MLPs** for different types of information captured by Mamba and the SWA layers



❖ SwiGLU : Swish + GLU (Gated Linear Unit) 결합

- LLM, Foundation Models에서 Activation Function으로 많이 사용

$$Swish(x) = x\sigma(\beta x)$$

- σ: Sigmoid Function $\sigma(x) = \frac{1}{1+e^{-x}}$
- β: 학습 가능한 파라미터

$$GLU(x, W, V, b, c) = \sigma(xW + b) \otimes (xV + c)$$

- x : Input
- W, V : 학습 가능한 텐서
- b, c : 학습 가능한 텐서 $bias$
- $\otimes$: Element-wise multiplication

$$SwiGLU(x, W, V, b, c, \beta) = Swish_\beta(xW + b) \otimes (xV + c)$$

Appendix C. IMPLEMENTATION DETAILS

Table 11: Detailed hyper-parameters of the baselines models trained on the Phi2 dataset with 230B tokens

Architecture	Llama-3	Mistral	Mamba	Mamba-SWA-MLP	Mamba-MLP
Parameters	1.6B	1.6B	1.8B	1.6B	1.9B
Batch size	2048	2048	2048	2048	2048
Learning rate	0.0006	0.0006	0.0006	0.0006	0.0006
Weight decay	0.1	0.1	0.1	0.1	0.1
Gradient clipping	1.0	1.0	1.0	1.0	1.0
Sequence length	4096	4096	4096	4096	4096
Sliding window size, w	-	2048	-	2048	-
Number of layers, N	48	48	64	54	48
Model width, d_m	2048	2048	2048	2048	2048
MLP intermediate size, d_p	8196	8196	-	8196	8196
Number of query heads	32	32	-	32	32
Number of KV heads	4	4	-	4	4
Number of Attention Layers	24	24	0	18	0
Number of Mamba Layers	0	0	64	18	24
Vocabulary size	50304	50304	50304	50304	50304

Table 12: Detailed hyper-parameters of the SAMBA models trained at different scales. We only show the optimization settings for the first training phase of the 3.8B model.

Total Parameters	421M	1.3B	1.7B	3.8B
Dataset	SlimPajama	SlimPajama	Phi-2	Phi-3
Batch size	512	512	2048	2048
Learning rate	0.0004	0.0004	0.0006	0.0006
Total training tokens	20B	100B	230B	3.2T
Weight decay	0.1	0.1	0.1	0.1
Gradient clipping	1.0	1.0	1.0	1.0
Sequence length	4096	4096	4096	4096
Sliding window size, w	2048	2048	2048	2048
Number of layers, N	24	36	48	64
Model width, d_m	1536	2304	2048	2816
MLP intermediate size, d_p	4096	6144	8196	9984
Number of query heads	12	18	32	11
Number of key-value heads	12	18	4	1
Vocabulary size	32000	32000	50304	32064

Experimental Results

3.1 LANGUAGE MODELING ON TEXTBOOK QUALITY DATA

Table 2: Downstream evaluation of the architectures trained on 230B tokens of the Phi2 dataset. We report the unnormalized accuracy for multiple choice tasks. GSM8K is evaluated with 5-shot examples while other tasks are in zero-shot. Best results are in bold, second best underlined.

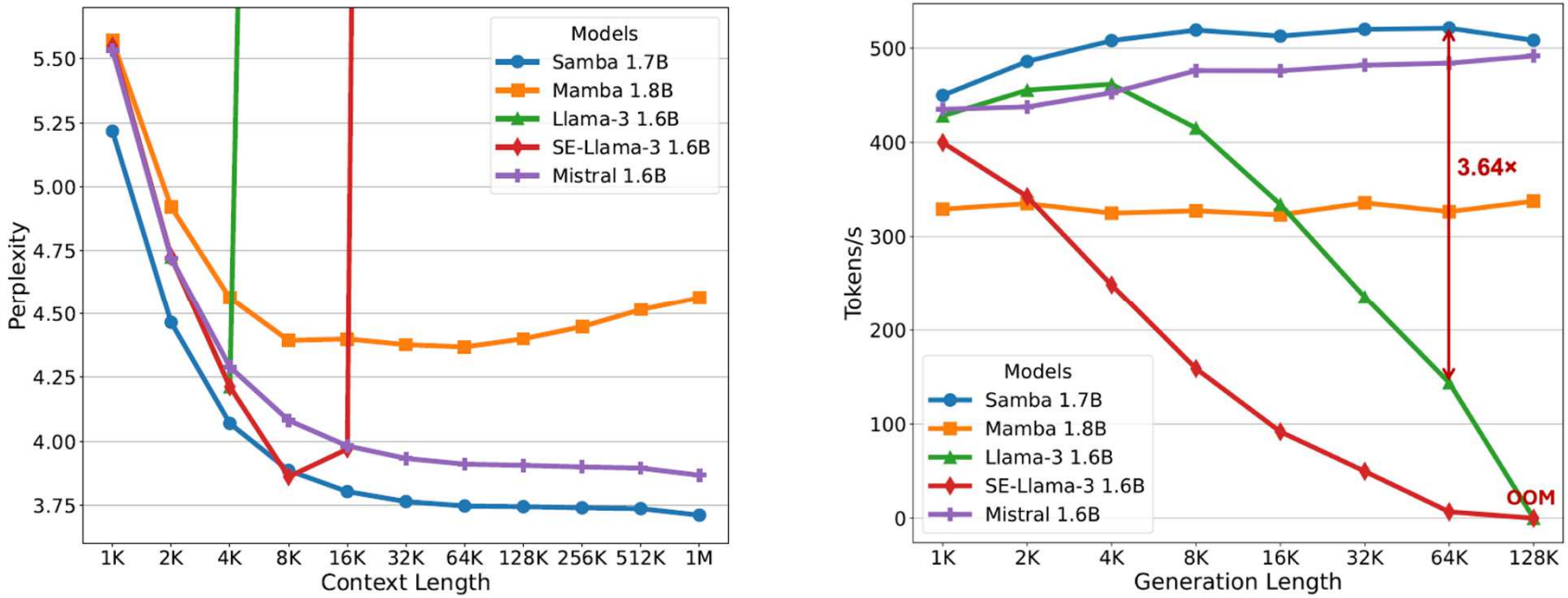
Benchmark	Llama-3 1.6B	Mistral 1.6B	Mamba 1.8B	Mamba-SWA-MLP 1.6B	Mamba-MLP 1.9B	SAMBA 1.7B
ARC-Easy	76.85	77.02	77.99	76.68	<u>78.91</u>	**79.25**
ARC-Challenge	43.26	44.20	45.22	46.16	<u>47.35</u>	**48.21**
PIQA	76.66	75.79	<u>77.31</u>	76.50	**78.84**	77.10
WinoGrande	70.01	70.72	<u>73.40</u>	**73.72**	72.38	72.93
SIQA	51.23	52.00	53.12	**55.12**	<u>54.30</u>	53.68
HellaSwag	46.98	47.19	<u>49.80</u>	49.71	**50.14**	49.74
BoolQ	68.20	70.70	<u>74.83</u>	74.74	73.70	**75.57**
OpenbookQA	34.00	32.80	<u>36.60</u>	33.80	35.40	**37.20**
SQuAD	74.88	72.82	67.66	<u>76.73</u>	63.86	**77.64**
MMLU	43.84	43.54	45.28	<u>47.39</u>	43.68	**48.01**
TruthfulQA (MC1)	25.70	25.09	26.81	26.20	<u>26.44</u>	**27.78**
TruthfulQA (MC2)	40.35	38.80	40.66	<u>40.80</u>	40.04	**41.62**
GSM8K	32.68	32.45	32.07	**44.05**	27.52	<u>38.97</u>
MBPP	46.30	47.08	<u>47.86</u>	47.08	47.08	**48.25**
HumanEval	36.59	36.59	35.98	<u>37.80</u>	31.10	**39.02**
Average	51.17	51.12	52.31	<u>53.77</u>	51.38	**54.33**

3.2 EXPLORATION ON HYBRIDIZING ATTENTION AND LINEAR RECURRENCE

Table 3: Perplexity on the validation set of SlimPajama for different attention and linear recurrent model architectures trained at 4,096 context length. We use window size 2,048 for Sliding Window Attention (SWA). The perplexity results have a fluctuation around ±0.3%.

Architecture	Size	Layers	Training Speed ($\times 10^5$ tokens/s)	Validation Context Length 4096	8192	16384
20B training tokens on 8×A100 GPUs						
Llama-2	438M	24	4.85	11.14	47.23	249.03
Llama-2-SWA	438M	24	4.96	11.12	10.66	10.57
Mamba	432M	60	2.46	10.70	10.30	10.24
Sliding GLA	438M	24	4.94	10.43	10.00	9.92
Sliding RetNet	446M	24	4.32	10.38	9.96	9.87
Mega-S6	422M	24	3.26	12.63	12.25	12.25
Mamba-SWA-MLP	400M	24	4.21	10.07	9.67	9.59
MLP2-SWA-MLP	417M	24	**5.08**	10.95	10.50	10.41
SAMBA-NoPE	421M	24	4.48	10.11	28.97	314.78
SAMBA	421M	24	4.46	**10.06**	**9.65**	**9.57**
100B training tokens on 64×H100 GPUs						
Llama-2	1.3B	40	25.9	7.60	44.32	249.64
Llama-2-SWA	1.3B	40	26.2	7.60	7.37	7.21
Mamba	1.3B	48	17.8	7.47	7.26	7.15
Sliding GLA	1.2B	36	25.9	7.58	7.35	7.19
Sliding RetNet	1.4B	36	23.0	7.56	7.35	7.56
Mega-S6	1.3B	36	17.9	9.01	8.81	8.68
Mamba-SWA-MLP	1.3B	36	23.5	7.37	7.16	7.00
MLP2-SWA-MLP	1.3B	36	**26.6**	7.81	7.58	7.42
SAMBA-NoPE	1.3B	36	25.2	7.33	20.40	326.17
SAMBA	1.3B	36	25.2	**7.32**	**7.11**	**6.96**

3.3 EFFICIENT LENGTH EXTRAPOLATION



(a) Perplexity on the test set of Proof-Pile

(b) Decoding throughput with batch size 16

Figure 2: SAMBA shows improved prediction up to 1M tokens in the Proof-Pile test set while achieving a 3.64× faster decoding throughput than the Llama-3 architecture on 64K generation length. We also include an SE-Llama-3 1.6B baseline which applies the SelfExtend (Jin et al., 2024) approach for zero-shot length extrapolation. All models are trained with 4K sequence length

3.4 LONG-CONTEXT UNDERSTANDING

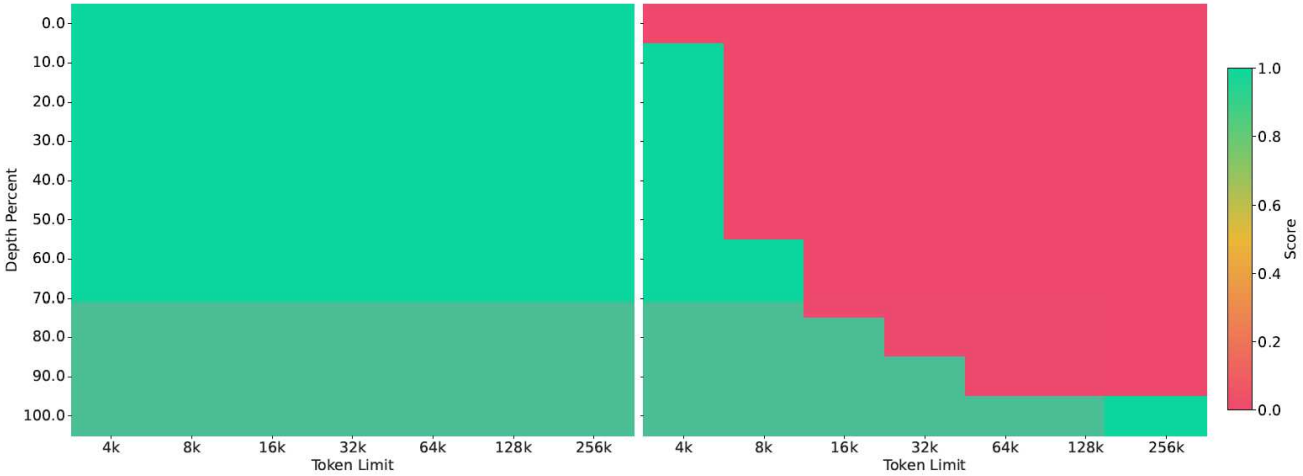


Figure 3: Passkey Retrieval performance up to 256K context length for SAMBA 1.7B (Left) vs. Mistral 1.6B (right) instruction tuned on 4K sequence length with 500 steps.

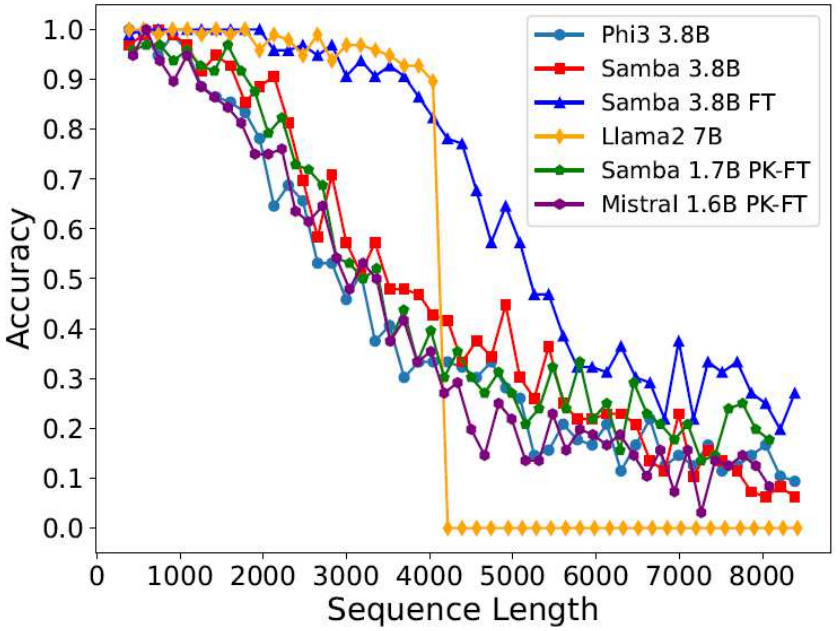


Figure 4: Phonebook evaluation accuracy of different base models

Analysis

Why not hybridize with full attention?

Table 5: Perplexity on SlimPajama of Mamba-MLP architectures with full attention layers replacing Mamba layers at different block indices. We define a block as two consecutive layers with a Mamba/Attention layer followed by an MLP. All the models have 12 blocks in total.

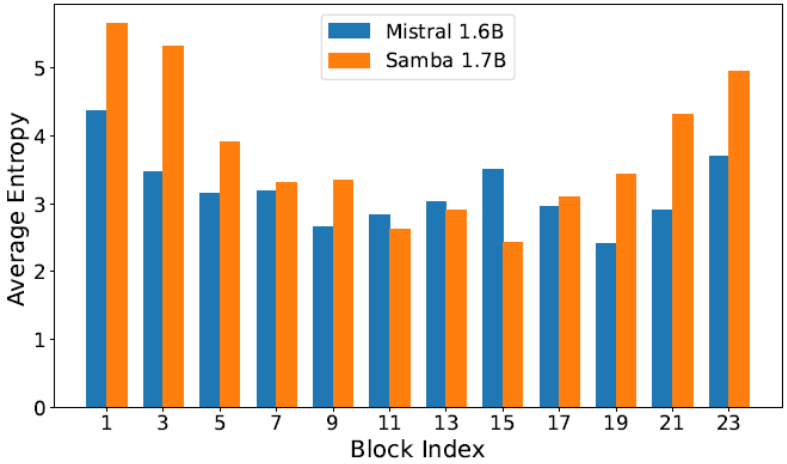
Architecture	Size	Block Index of Full Attention	Training Speed ($\times 10^5$ tokens/s)	Validation Context Length 4096	8192	16384
Mamba-MLP	449M	11	7.78	10.29	10.53	13.66
	449M	5	7.78	10.10	10.05	12.83
	449M	0	7.78	10.89	10.55	10.63
	443M	1, 5	7.93	**10.06**	10.34	13.57
SAMBA	421M	SWA at odd indices	8.59	**10.06**	**9.65**	**9.57**

How many parameters should be allocated to Attention?

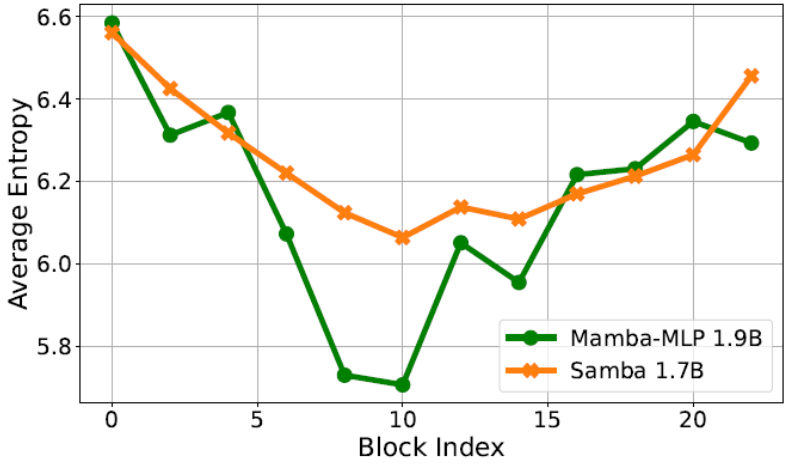
Table 6: Perplexity on SlimPajama of Llama-2-SWA and Samba models at the 430M scales trained with different number of Query and Key-Value heads. "KV Size" means the size of Key-Value vectors per token and attention layer. Since grouped query attention will reduce the parameters for attention from $4d_m^2$ to roughly $2d_m^2$, we increase the intermediate size of MLP from $8/3d_m$=4608 to have roughly the same number of total parameters as the original models.

Query Head	Key-Value Head	Head Dim.	KV Size	Model Size	Training Speed ($\times 10^5$ tokens/s)	Validation Context Length 4096	8192	16384
Llama-2-SWA Architecture								
12	2	128	512	419M	10.01	11.11	10.64	10.56
6	1	256	512	419M	9.98	11.09	10.62	10.54
12	1	128	256	414M	10.25	**10.89**	**10.44**	**10.35**
12	4	128	1024	428M	9.85	11.11	10.64	10.56
Samba Architecture								
12	2	128	512	426M	8.55	10.09	9.68	9.60
6	1	256	512	426M	8.46	**9.99**	**9.59**	**9.51**
12	1	128	256	424M	8.62	10.07	9.66	9.58
12	4	128	1024	431M	8.57	10.02	9.62	9.55

Analysis



(a) Average attention entropy per decoding step



(b) Average S6 selection entropy on full sequences

Figure 5: The average entropy of the attention mechanism and the Mamba's S6 input selection mechanism at each block of layers on 100 random samples from the GSM8K dataset.