MossNet: Mixture of State-Space Experts is a Multi-Head Attention

Shikhar Tuli, James Smith, Haris Jeelani, Chi-Heng Lin, Abhishek Patel, Vasili Ramanishka, Yen-Chang Hsu, Hongxia Jin
Samsung Research America
665 Clyde Ave, Mountain View, CA 94043

Abstract

Large language models (LLMs) have significantly advanced generative applications in natural language processing (NLP). Recent trends in model architectures revolve around efficient variants of transformers or state-space/gated-recurrent models (SSMs, GRMs). However, prevailing SSM/GRM-based methods often emulate only a single attention head, potentially limiting their expressiveness. In this work, we propose **MossNet**, a novel mixture-of-state-space-experts architecture that emulates a linear multi-head attention (MHA). MossNet leverages a mixture-of-experts (MoE) implementation not only in channel-mixing multi-layered perceptron (MLP) blocks but also in the time-mixing SSM kernels to realize multiple "attention heads." Extensive experiments on language modeling and downstream evaluations show that MossNet outperforms both transformer- and SSM-based architectures of similar model size and data budgets. Larger variants of MossNet, trained on trillions of tokens, further confirm its scalability and superior performance. In addition, real-device profiling on a Samsung Galaxy S24 Ultra and an Nvidia A100 GPU demonstrate favorable runtime speed and resource usage compared to similarly sized baselines. Our results suggest that MossNet is a compelling new direction for efficient, high-performing recurrent LLM architectures.

1 Introduction

Rapid advancements in training and deployment of foundation models have revolutionized various generative applications, including the development of sophisticated chatbots (OpenAI, 2024a), generation of video (OpenAI, 2024b), coding assistance (Roziere et al., 2023), and robotic manipulation (Brohan et al., 2023). With an increasing number of LLM architectures being proposed, such as transformers (Vaswani et al., 2017; Brown et al., 2020), SSMs (Gu et al., 2021, 2022), and linear GRMs (Katsch, 2023; Qin et al., 2024)[1], the field of NLP continues to evolve at a remarkable pace. The continuous development of these models presents both opportunities and challenges.

1.1 Challenges and Motivation

Transformers, introduced by Vaswani et al. (2017), have been particularly influential in NLP due to their success in language modeling. The transformer architecture relies on a stack of MHA and MLP blocks. Despite their effectiveness, transformers face several efficiency challenges, including an inability to model outside the context window (although, recent works attempt to mitigate this; Munkhdalai et al. 2024), quadratic scaling of compute, and linear scaling of cache with respect to context length. Efficient variants have been proposed that attempt to overcome these drawbacks (Tay et al., 2022). Other recent works aim to improve efficiency by replacing the MLP block with a mixture-of-expert (MLP-MoE) block (Fedus et al., 2022; Jiang et al., 2024) or the MHA block with a mixture-of-attention (MHA-MoA) block (Zhang et al., 2022). However, these solutions often trade trade performance for efficiency.

SSMs, along with recently proposed GRMs, present a promising alternative to the transformer architecture, offering better computational and memory efficiency due to their inherent recurrent design. Gu and Dao (2023) introduced Mamba, a hardware-optimized selective SSM that achieves high efficiency without sacrificing performance, thanks to the work-efficient parallel scan algorithm (Blelloch, 1990; Martin and Cundy, 2018). Recently proposed extensions of the Mamba architecture, including BlackMamba/MoE-Mamba (Anthony et al., 2024; Pióro et al., 2024) and Jamba (Lieber et al., 2024), along with other

[1]Although SSMs can be considered a specific subset of GRMs, we distinguish them due to their distinct terminology in the literature and their basis in state-space theory, encompassing both continuous-time systems and their discretization.

parallely-proposed GRMs (Peng et al., 2023; Sun et al., 2023; Katsch, 2023; De et al., 2024) match the performance of transformers while maintaining the benefits of recurrent models. More importantly, these works show that such models can emulate the self-attention operation in their mathematical parallel formulation, albeit only a single attention head.

1.2 Our Contribution

Due to the single attention head modeling in existing SSMs and GRMs, they exhibit many performance drawbacks (Jelassi et al., 2024; Lieber et al., 2024; Patro and Agneeswaran, 2024). Hence, in this work, we propose MossNet, a robust and scalable alternative to current LLM architectures based on a mixture of state-space experts, addressing key challenges, and pushing the boundaries of what is achievable with SSMs. MossNet attempts to model an MHA by extending an existing SSM architecture. More concretely, we summarize the contributions of this work next.

- We propose MossNet, a novel architecture that models not just a single self-attention head but an MHA (specifically, its linear mixture-of-expert implementation, i.e., MHA-MoA), just like state-of-the-art transformer models (with linear attention). We mathematically show how a mixture of state-space experts models an MHA.

- We do a *fair* comparison of recently-proposed LLM architectures based on perplexity (PPL) and downstream benchmark performance for small-scale models. Through rigorous experimentation, we empirically show how MossNet outperforms other popular transformer- and SSM/GRM-based baselines.

- We train larger variants of MossNet models, namely MossNet-8x200M+, and compare it against state-of-the-art baselines of similar active and total parameter counts. MossNet-8x200M+, in top-2 mode, outperforms Qwen2.5-0.5B by a significant margin, despite being trained on a fraction of pre-training tokens.

- We profile the prefill and generation speed of the proposed MossNet models on a Samsung Galaxy S24 Ultra smartphone and an Nvidia A100 GPU. On resource-constrained devices, MossNet-8x200M+ is *significantly* faster in terms of prefill and generation speed along with memory consumption when compared to transformers- and SSM-based baselines with similar active parameter counts.

The rest of the article is organized as follows. Section 2 details the MossNet architecture along with the proposed evaluation methods. Section 3 presents the experimental results. Finally, Section 4 concludes the article and Section 5 provides the limitations.

2 Method

In this section, we discuss the implementation details of the MossNet model.

2.1 Preliminaries

We now discuss the required background on the Mamba architecture and the traditional MoE implementation in models like Mixtral-8x7B (Jiang et al., 2024) and BlackMamba/MoE-Mamba (Anthony et al., 2024; Pióro et al., 2024).

2.1.1 Mamba

SSMs are a class of sequence models with linear complexity with respect to the sequence length. This results in superior efficiency, especially for long-context input. Multi-dimensional SSMs are defined using four parameters $\boldsymbol{\Delta}$, $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$, and sequence-to-sequence transformations from $\mathbf{x}_t \in \mathbb{R}^N$ to $\mathbf{y}_t \in \mathbb{R}^M$ through an implicit latent state $\mathbf{s}_t \in \mathbb{R}^P$ as follows (Gu et al., 2022),

$$\mathbf{s}'_t = \boldsymbol{A}\mathbf{s}_t + \boldsymbol{B}\mathbf{x}_t \quad (1)$$
$$\mathbf{y}_t = \boldsymbol{C}\mathbf{s}_t \quad (2)$$

where, $\boldsymbol{A} \in \mathbb{R}^{P\times P}$, $\boldsymbol{B} \in \mathbb{R}^{P\times N}$, $\boldsymbol{C} \in \mathbb{R}^{M\times P}$, and $\mathbf{s}'_t$ is the derivative of $\mathbf{s}_t$. In its discrete parameterization,

$$\mathbf{s}_t = \bar{\boldsymbol{A}}\mathbf{s}_{t-1} + \bar{\boldsymbol{B}}\mathbf{x}_t \quad (3)$$
$$\mathbf{y}_t = \boldsymbol{C}\mathbf{s}_t \quad (4)$$

where,

$$\bar{\boldsymbol{A}} = \exp(\boldsymbol{\Delta A}), \text{and} \quad (5)$$
$$\bar{\boldsymbol{B}} = (\boldsymbol{\Delta A})^{-1}(\exp(\boldsymbol{\Delta A}) - \boldsymbol{I}) \cdot \boldsymbol{\Delta B}. \quad (6)$$

One can efficiently compute a linear dynamical system like this in parallel via a convolution (Gu et al., 2022) or parallel associative scan (Blelloch, 1990). On the other hand, one can leverage the recurrent form presented above for rapid generation at inference time. Mamba (Gu and Dao, 2023) makes the discrete parameters input-dependent, i.e., $\bar{\boldsymbol{A}}_t$, $\bar{\boldsymbol{B}}_t$, and $\boldsymbol{C}_t$.

Gu and Dao (2023) offer an intuitive interpretation of these parameters. $\bar{\boldsymbol{A}}$ controls the transition dynamics, while $\bar{\boldsymbol{B}}$ and $\boldsymbol{C}$ control the selectivity of the input x_t into the hidden state h_t and the state into the output y_t, respectively. Finally, $\boldsymbol{\Delta}$ controls the balance between how much to focus or ignore the current input x_t. However, in an MHA, each head focuses on different aspects of the relationships between words/tokens. An MHA thus provides enhanced expressiveness, mitigates information loss, and improves learning capability compared to a single attention head. In the *same* spirit, we hypothesize that in an SSM, there should be multiple such parameters that focus on different parts of the input sequence. For instance, multiple $\boldsymbol{\Delta}$'s could focus on the selectivity of the current input in the context of multiple dependencies in data.

2.1.2 Mixture of Experts

Primarily, MoEs are synonymous with MLP layers within a transformer model (we call this the MLP-MoE block; Fedus et al. 2022). Such models reduce the inference cost by routing tokens to specific MLP *experts*. A router maps the token representations to experts, where each expert is simply a standard transformer MLP block. The expert to whom the token is routed is chosen from the top-k of the expert probabilities, where k is a hyperparameter. Mathematically, an input $\mathbf{x}_t$ is mapped through the router to a probability distribution $p_i(\mathbf{x}_t)$, where i labels the experts. Upon selecting the top-k probabilities, the output of the MoE layer at time-step t, i.e., $\mathbf{y}_t$ is a linearly weighted combination of each expert's computation on the input,

$$\mathbf{y}_t = \sum_{i \in \text{top-}k} p_i(\mathbf{x}_t)\boldsymbol{E}_i(\mathbf{x}_t) \tag{7}$$

where $\boldsymbol{E}_i$ is the i-th MLP expert.

Instead of applying MoE to the, channel-mixing, MLP layers, Zhang et al. (2022) apply the MoE to the, time-mixing, MHA blocks (we call this the MHA-MoA block). This block performs as well as the traditional MHA, while providing the benefits of MoE (Fedus et al., 2022). We take inspiration from the MHA-MoA block in order to emulate multiple *attention* heads in the proposed MossNet architecture.

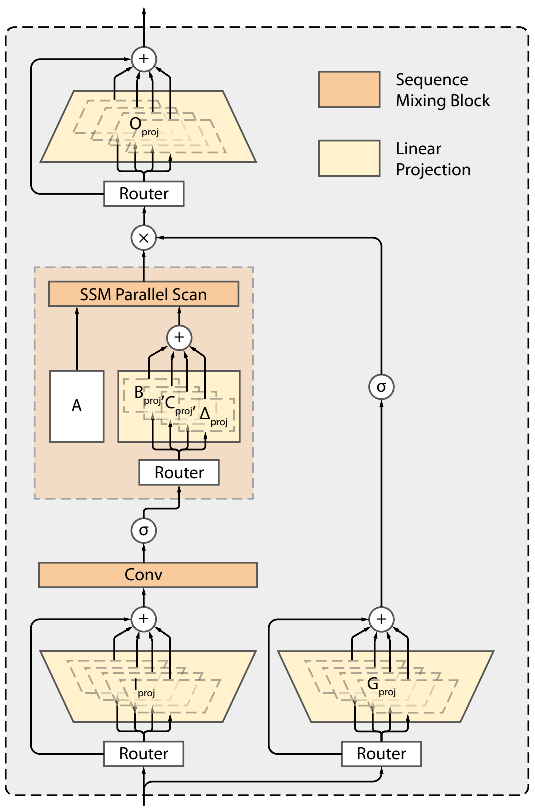

Figure 1: Simplified working schematic of the MossNet block. We implement MoE in channel mixing input, gate, and output projections and time mixing input-dependent SSM parameters $\boldsymbol{B}$, $\boldsymbol{C}$, and $\boldsymbol{\Delta}$.

2.2 MossNet Architecture

MossNet extends the Mamba architecture (Gu and Dao, 2023) by implementing MoE in various projection operations. Fig. 1 shows a working schematic of the MossNet block. Concretely, we implement MoE for the channel-mixing linear projections ($\boldsymbol{I}$, $\boldsymbol{G}$, and $\boldsymbol{O}$) and the sequence transformation input-dependent SSM parameters $\boldsymbol{B}$, $\boldsymbol{C}$, and $\boldsymbol{\Delta}$. The input-independent parameter $\boldsymbol{A}$, along with $\boldsymbol{B}$ and $\boldsymbol{\Delta}$, are used to calculate the discrete SSM parameters $\bar{\boldsymbol{A}}$ and $\bar{\boldsymbol{B}}$. The combined contribution of the mixture of state-space experts is input to the hardware-optimized SSM parallel scan kernel (Gu and Dao, 2023).

We follow Fedus et al. (2022) to implement the router network for the MoE implementation. Concretely, the router (implemented as a feed-forward layer) calculates the score $h(\mathbf{x}_t) \in \mathbb{R}^{N_\text{experts}}$, where N_experts is the number of experts. We normalize the scores using a softmax operation to obtain $p_i(\mathbf{x}_t)$ in Eq. (7). For equiproportionate distribution of tokens to the experts, we employ the load balancing loss (Fedus et al., 2022) and add it to the training objective with a weight α.

Theorem 1. *A mixture-of-expert implementation of $\bar{\boldsymbol{A}}$, $\bar{\boldsymbol{B}}$, and $\boldsymbol{C}$ is equivalent to a mixture-of-expert implementation of a linear multi-head attention.*

Proof. Recall that the state evolution of a discretely parameterized multi-dimensional selective SSM is

$$\mathbf{s}_t = \bar{\boldsymbol{A}}_t \mathbf{s}_{t-1} + \bar{\boldsymbol{B}}_t \mathbf{x}_t \tag{8}$$

$$\mathbf{y}_t = \boldsymbol{C}_t \mathbf{s}_t. \tag{9}$$

Expanding Eq. (9), we get

$$\begin{aligned}\mathbf{y}_t &= \sum_{i=1}^{t} \boldsymbol{C}_t \prod_{j=i+1}^{t} \left(\bar{\boldsymbol{A}}_j\right) \bar{\boldsymbol{B}}_i \mathbf{x}_i \\ &= \sum_{i=1}^{t} \left(\boldsymbol{C}_t \prod_{j=1}^{t} \bar{\boldsymbol{A}}_j\right) \left(\prod_{j=1}^{i} \bar{\boldsymbol{A}}_j^{-1} \bar{\boldsymbol{B}}_i\right) \mathbf{x}_i. \quad (10)\end{aligned}$$

On the other hand, the mixture-of-expert implementations of $\bar{\boldsymbol{B}}_t$, and $\boldsymbol{C}_t$ can be written as,

$$\bar{\boldsymbol{B}}_t = \sum_{m=1}^{N_{\text{experts}}} p_m(\mathbf{x}_t) \bar{\boldsymbol{B}}_t^m, \tag{11}$$

$$\boldsymbol{C}_t = \sum_{n=1}^{N_{\text{experts}}} p_n(\mathbf{x}_t) \boldsymbol{C}_t^n, \tag{12}$$

where the experts are functions of input $\mathbf{x}_t$. Now, plugging in Eqs. (11) and (12) into Eq. (10), we obtain the output at time t,

$$\begin{aligned}\mathbf{y}_t = \sum_{m,n=1}^{N_{\text{experts}}} \sum_{i=1}^{t} &\left(p_m(\mathbf{x}_t) \boldsymbol{C}_t^m \prod_{j=1}^{t} \bar{\boldsymbol{A}}_j\right) \times \\ &\left(p_n(\mathbf{x}_t) \prod_{j=1}^{i} \bar{\boldsymbol{A}}_j^{-1} \bar{\boldsymbol{B}}_i^n\right) \mathbf{x}_i. \quad (13)\end{aligned}$$

If we define

$$\mathbf{q}_t^m = p_m(\mathbf{x}_t) \boldsymbol{C}_t^m \left(\prod_{j=1}^{t} \bar{\boldsymbol{A}}_j\right),$$

$$\mathbf{k}_i^n = p_n(\mathbf{x}_t) \left(\prod_{j=1}^{i} \bar{\boldsymbol{A}}_j^{-1}\right) \bar{\boldsymbol{B}}_i^n, \mathbf{v}_i = \mathbf{x}_i,$$

then we can put the expression of the output into a form of a weighted, linear MHA-MoA:

$$\mathbf{y}_t = \sum_{m,n=1}^{N_{\text{experts}}} \sum_{i=1}^{t} \langle \mathbf{q}_t^m, \mathbf{k}_i^n \rangle \mathbf{v}_i = \sum_{m,n=1}^{N_{\text{experts}}} \text{Attention}_{m,n},$$

where we interpret $\mathbf{q}^m$, $\mathbf{k}$, and $\mathbf{v}$ as the m-th head's query vector, the n-th head's key vector, and the shared value vector for all heads, respectively. Finally, we remark that the above expression does not use an output projection since the value vector is shared and equal to $\mathbf{x}_t$ for all heads. □

Remark 1 The above expression differs from the traditional MHA in three aspects: 1) Each head's query interacts with the keys from all other heads, contrasting with the standard approach where queries interact only with their corresponding keys. 2) The key and query are functions of the router probabilities, making them non-linear functions of the input. 3) The value vector is shared among all heads, which eliminates the need for an output projection.

Remark 2 The above expression differs from the MHA-MoA implementation by Zhang et al. (2022) in that the router leverages multiple query and key experts, instead of multiple query experts and common key/value experts.

Remark 3 In the above formulation, we neglect the MoE implementation of $\bar{\boldsymbol{A}}$, a function of $\boldsymbol{A}$ and $\boldsymbol{\Delta}$, for simplicity. In MossNet, we implement $\boldsymbol{\Delta}$ as an MoE as well. This would be equivalent to the above formulation, however, at the cost of a more complex set of equations.

2.3 Training and Evaluation Setup

To *fairly* compare different architectures, we train a suite of models with varying number of parameters on the same language modeling dataset. Concretely, we compare the performance of various architectures. These include three popular transformer architectures: Pythia (Biderman et al., 2023), Llama (Touvron et al., 2023), Mistral (Jiang et al., 2023) and its MoE extension Mixtral (Jiang et al., 2024), a recently-proposed GRM, i.e., Griffin (De et al., 2024), along with Mamba (Gu and Dao, 2023) and its extensions: Zamba (Glorioso et al., 2024) and MoE-Mamba (Anthony et al., 2024; Pióro et al., 2024). We also add recently-proposed Mamba2 (Dao and Gu, 2024) to our comparisons. We use the same BPE tokenizer for all models (Black et al., 2022). We train these models on the Cosmopedia (Ben Allal et al., 2024a) dataset, which has shown high model performance per pre-training token. We present additional model hyperparameters along with other training details in Appendix A.

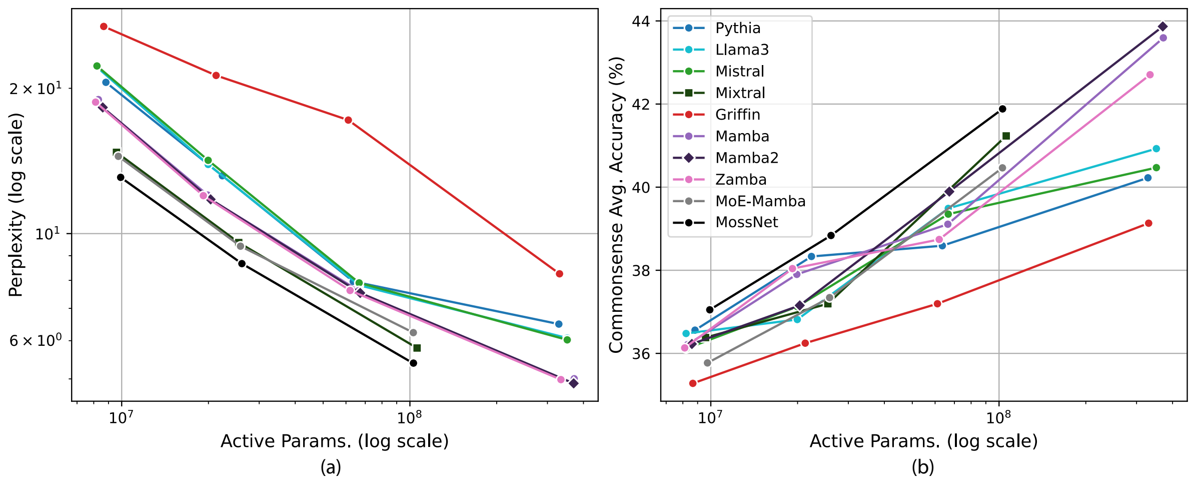

Figure 2: (a) Perplexity and (b) commonsense average accuracy scaling for *fairly-trained* models.

Following the *fairly-trained* setting, we train a larger model, namely MossNet-8x200M+, on a custom dataset with 2.8T tokens comprised of a mixture of existing open-source datasets. We describe the hyperparameter choices and the training recipes employed for this model in Appendix A. We provide details of the custom pre-training dataset in Appendix B.

We compare the performance of the proposed MossNetsuite of model against baselines on various downstream benchmarks.

3 Experiments

In this section, we present experimental results comparing the proposed MossNet suite of models against *fairly-trained* and state-of-the-art baselines.

3.1 Downstream Language Modeling Performance

First, we evaluate the MossNet architecture, along with other baselines, based on language modeling perplexity on the Cosmopedia dataset and consider eight standard commonsense reasoning benchmarks: ARC challenge (ARC-c) and ARC easy (ARC-e, Clark et al. 2018), BoolQ (Clark et al., 2019), COPA (Roemmele et al., 2011), HellaSwag (Zellers et al., 2019), OpenBookQA (OBQA, Mihaylov et al. 2018), PIQA (Bisk et al., 2020), and WinoGrande (Sakaguchi et al., 2021). We perform evaluations in the zero-shot setting as done in the language modeling community. We *fairly* train all models on the same dataset and under the same setting (more details in Appendix A).

Fig. 2 shows how the performance scales for MossNet and other baselines, both dense and sparse. MossNet achieves lower perplexity and higher average commonsense accuracy, showing superior scaling across model sizes. This shows the advantages of multiple state-space “heads” in language modeling performance.

We also evaluate MossNet and other baselines on more benchmarks: infromation retrieval on SWDE and FDA (Arora et al., 2024), closed-book question answering on TriviaQA (Joshi et al., 2017), reading comprehension on SQuADv2 (Rajpurkar et al., 2018) and RACE (Lai et al., 2017), and general knowledge and reasoning on MMLU (Hendrycks et al., 2021). Table 1 shows the results. MossNet outperforms baselines with similar number of active parameters on most benchmarks.

Finally, we train MossNet-8x200M+ for 2.8T tokens on a custom pretraining dataset and compare it against state-of-the-art baselines. We trained MossNet-8x200M+ to support both top-2 and top-3 modes, resulting in 477M and 657M active parameters, respectively (more details in Section A). This allows the same model to support low-power and high-power models on-device. Table 2 shows the results. In the top-2 mode, we compare MossNet with Mamba-370M (Gu and Dao, 2023), Mamba2-370M (Dao and Gu, 2024), BlackMamba-1.5B (Anthony et al., 2024), Hymba-350M (Dong et al., 2024), Qwen2.5-0.5B (Yang et al., 2024), and SmolLM2-360M (Allal et al., 2024). MossNet-8x200M+ outperforms Qwen2.5-0.5B by 5.8% average accuracy. In the top-3 mode, we compare MossNet with Mamba-790M, Mamba2-790M, and BlackMamba-2.8B. We also see notable improvement going from top-2 to top-3 gating. Thanks to the proposed MoE design and training strat-

Table 1: Performance of MossNet and other baselines on SWDE (zero-shot accuracy), FDA (zero-shot accuracy), TriviaQA (closed-book, zero-shot accuracy), SQuADv2 (zero-shot F1), RACE (zero-shot accuracy) and MMLU (five-shot accuracy) benchmarks.

Model	**Recall**		**Closed-book**	**Reading Comprehension**		**MMLU**				**Avg.**
	SWDE	**FDA**	**TriviaQA**	**SQuADv2**	**RACE**	**Hum.**	**Social Sci.**	**STEM**	**Other**	
Pythia-9M	0.9	0.0	0.0	11.1	23.1	24.7	22.8	23.2	23.4	14.4
Llama-8M	1.4	**0.2**	0.0	4.2	23.6	24.9	23.8	**26.9**	25.6	14.5
Mistral-8M	0.8	0.1	0.0	11.8	24.3	24.5	24.0	22.9	24.2	14.7
Mixtral-8x8M	0.1	0.0	0.1	15.5	23.4	23.9	21.7	22.4	25.4	14.7
Griffin-9M	1.1	0.0	0.0	20.6	22.5	24.2	21.7	21.3	24.0	15.0
Mamba-8M	0.6	0.0	0.0	9.8	24.0	24.5	22.0	22.7	23.6	14.1
Mamba2-9M	1.3	0.1	**0.1**	6.8	**25.4**	24.1	22.4	22.3	24.8	14.1
Zamba-8M	**2.0**	0.1	0.1	31.1	23.2	23.9	24.6	26.9	25.2	17.5
MoE-Mamba-8x8M	1.1	0.0	0.0	**37.8**	22.8	24.4	23.1	23.0	24.5	17.4
MossNet-8x8M	1.4	**0.2**	0.0	34.7	24.4	**25.1**	**25.0**	24.9	**25.8**	**17.9**
Pythia-22M	0.8	0.3	0.0	21.6	25.4	25.4	23.4	27.4	24.1	16.5
Llama-20M	2.3	0.1	0.1	15.2	24.4	24.1	22.4	23.9	23.3	15.1
Mistral-20M	0.7	0.0	0.1	5.5	24.2	23.9	21.9	23.0	23.8	13.7
Mixtral-8x20M	1.1	0.0	0.3	5.5	23.6	24.8	23.8	25.0	24.4	14.3
Griffin-22M	0.9	0.1	0.0	25.5	21.2	24.2	21.9	22.5	23.6	15.5
Mamba-20M	0.9	0.1	0.2	6.1	22.7	24.2	22.5	22.0	23.8	13.6
Mamba2-20M	0.9	0.2	0.1	4.8	25.6	24.6	23.8	27.6	24.0	14.8
Zamba-20M	4.7	**0.5**	0.1	9.2	24.3	24.0	23.3	25.8	27.8	15.5
MoE-Mamba-8x20M	3.1	0.0	0.4	1.9	25.3	**26.3**	24.3	25.3	24.4	14.7
MossNet-8x20M	**4.8**	0.3	**0.4**	**26.3**	**25.8**	24.3	**26.0**	**28.8**	**27.9**	**20.1**
Pythia-64M	5.6	0.5	0.8	21.3	27.6	24.8	24.6	**28.4**	24.5	16.5
Llama-67M	7.8	0.4	0.8	13.3	25.8	24.1	24.6	26.2	25.8	16.5
Mistral-67M	0.3	0.0	0.6	5.0	24.0	24.1	23.2	26.4	22.7	14.0
Mixtral-8x67M	0.5	0.0	2.2	16.1	26.0	24.4	24.5	22.3	22.5	15.4
Griffin-61M	0.5	0.0	0.0	32.3	23.6	24.3	22.0	21.5	23.8	16.4
Mamba-66M	3.7	0.3	0.7	3.7	25.6	25.1	23.5	25.0	23.2	14.5
Mamba2-67M	3.3	0.2	0.5	2.5	25.6	**25.4**	24.4	25.9	25.5	14.8
Zamba-62M	8.3	0.4	1.2	3.3	25.7	24.7	23.8	26.9	25.5	15.5
MoE-Mamba-8x66M	5.0	0.6	1.1	3.0	25.9	25.1	23.4	25.0	23.1	14.7
MossNet-8x66M	**13.0**	**1.4**	**2.9**	**34.4**	**27.9**	25.2	**24.8**	25.4	**25.9**	**20.1**
Pythia-330M	11.0	0.5	1.1	3.0	26.7	25.1	28.6	27.6	24.1	16.4
Llama-350M	9.3	0.7	1.4	4.3	26.8	24.8	**30.5**	28.1	24.4	16.7
Mistral-350M	7.9	0.0	2.1	3.2	27.6	**26.6**	25.0	26.8	23.9	15.9
Griffin-330M	3.5	0.1	0.2	**13.8**	23.4	25.4	24.6	27.1	24.9	15.9
Mamba-370M	4.8	0.4	1.7	4.4	27.1	25.8	25.2	**28.2**	24.1	15.7
Mamba2-370M	8.2	0.8	1.8	3.4	**28.9**	24.4	22.6	23.1	25.4	15.4
Zamba-330M	**19.7**	**6.4**	**2.4**	9.1	27.9	24.1	29.5	26.5	**25.7**	**19.0**

egy, MossNet exhibits flexibility in different active-parameter-constrained settings, unlike static models. We also present the performance of state-of-the-art models with active parameters around 1.5B. MossNet not only outperforms baselines with similar active parameters, but also reaches the performance of other models with 1.5B active parameters, while achieving *significant* latency and memory gains as we show next.

3.2 Speed and Memory Profiling

In this section we present the memory and speed profiling results on server (Nvidia A100-80GB GPU) and on mobile (Samsung Galazy S24 Ultra).

3.2.1 Server GPU Results

Fig. 3 presents (a) memory consumption, (b) prefill speed, and (c) generation speed across increasing context lengths for MossNet-8×200M+ compared to several single-expert baselines of similar active and total parameter scale. While all models naturally require more GPU memory as context length grows, MossNet's MoE design contains that growth more effectively, keeping memory usage lower than monolithic baselines with comparable or larger parameter counts. Particularly, for longer contexts, e.g. 32K, MossNet-8x200M+ in top-2 mode achieves the lowest memory usage relative to baselines. MossNet also demonstrates consistently

Table 2: Performance of MossNet and state-of-the-art baselines on ARC-c (zero-shot accuracy), ARC-e (zero-shot accuracy), HellaSwag (zero-shot accuracy), PIQA (zero-shot accuracy), WinoGrande (zero-shot accuracy), SQuADv2 (zero-shot F1 score), and MMLU (five-shot accuracy) benchmarks. We evaluate the instruction-tuned models wherever available. *We evaluate all models except Hymba-350M (not publicly available) using `lm-evaluation-harness` (Gao et al., 2024). For Hymba-350M, we present the reported results (Dong et al., 2024).

	Model	**Train Tokens**	**ARC-c**	**ARC-e**	**HellaSwag**	**PIQA**	**WinoGrande**	**SQuADv2**	**MMLU**	**Average**
~500M	Mamba-370M	0.3T	28.0	55.1	46.5	69.5	55.3	5.8	23.1	40.5
	Mamba2-370M	0.3T	26.9	54.9	46.9	70.5	55.7	5.9	23.6	40.6
	BlackMamba-1.5B	0.3T	24.1	56.1	36.5	69.0	52.6	4.7	19.4	37.5
	Hymba-350M*	1.5T	-	-	55.1	72.9	57.8	-	34.5	-
	Qwen2.5-0.5B	18T	34.2	59.8	53.0	70.9	56.5	12.3	**47.4**	47.7
	SmolLM2-360M	11T	34.3	49.6	57.3	72.0	57.8	9.3	26.2	43.8
	MossNet-8x200M+ (top-2)	2.8T	**41.4**	**68.4**	**63.9**	**76.1**	**62.5**	**20.9**	41.2	**53.5**
~700M	Mamba-790M	0.3T	29.5	61.2	55.1	72.1	56.1	8.5	24.0	43.8
	Mamba2-790M	0.3T	28.5	61.0	54.9	72.0	60.2	8.7	24.4	44.2
	BlackMamba-2.8B	0.3T	24.5	60.3	39.7	71.2	52.1	6.8	22.7	39.6
	MossNet-8x200M+ (top-3)	2.8T	**40.2**	**69.8**	**65.9**	**76.3**	**64.2**	**28.1**	**43.6**	**55.4**
~1.5B	Rene-1.3B	1.5T	34.4	61.7	69.4	77.5	62.9	17.4	32.6	50.8
	Hymba-1.5B	1.5T	44.3	76.0	71.1	77.4	66.6	20.3	52.3	58.3
	Phi1.5	0.15T	48.5	73.9	62.6	76.4	73.6	19.4	42.9	56.8
	DCLM-1.4B	4.3T	47.5	75.7	73.2	78.5	67.5	28.9	51.6	60.4
	Qwen2.5-1.5B	18T	47.0	76.7	69.0	76.8	63.5	25.2	60.9	59.9
	SmolLM2-1.7B	11T	44.1	63.6	72.6	77.0	69.1	19.3	49.9	56.5

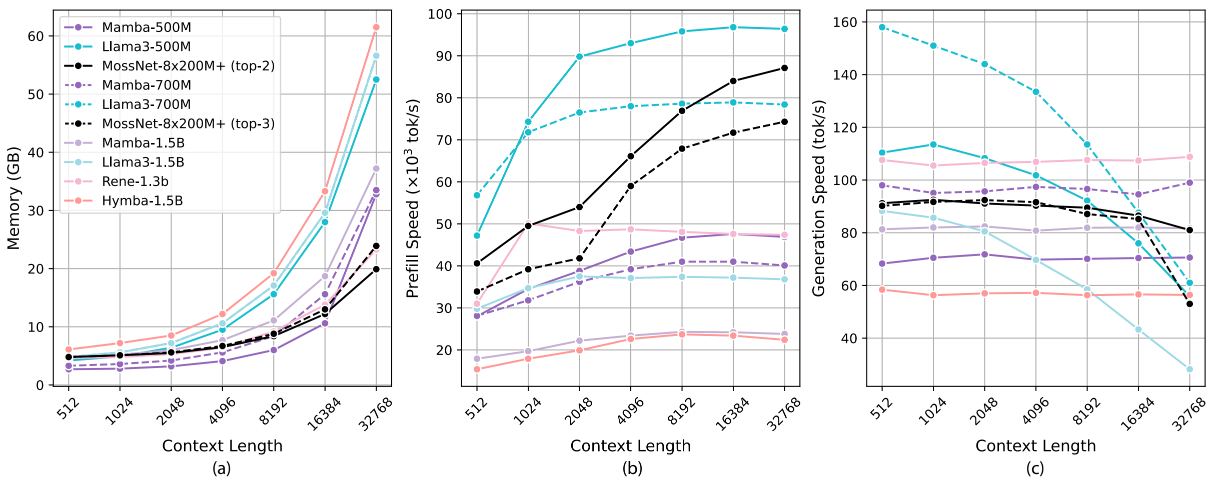

Figure 3: (a) Memory consumption, (b) prefill speed, and (c) generation speed with context length for MossNet-8x200M+ and baselines on A100-80GB (FP16 precision, FlashAttention 2). Batch size set to 4.

high prefill throughput. Its prefill speed approaches that of Llama3-500M/700M, being far superior to other SMM/hybrid baselines. Further, as shown in Fig 3(c), MossNet's token-by-token generation speed remains stable across large contexts, whereas competing baselines often slow down significantly. In short, these GPU-based results highlight the key advantages of expert routing: more efficient memory usage and stronger large-context performance, without sacrificing speed.

3.2.2 Mobile Results

Fig. 4 illustrates the same three metrics on a Samsung Galaxy S24 Ultra (on CPU with Q8 precision) for a batch size of 1, further underscoring MossNet's benefits in resource-constrained edge settings. Here, MossNet's memory footprint stays essentially flat at around 1.6 GB across all context lengths, while the Llama3 models consume increasingly large amounts of memory as the context grows. Mamba too has a flat memory curve due to serial operation of the scan operation on-device. MossNet's prefill and generation speeds remain comfortably higher and more consistent than those of baselines, which degrade more severely as context length increases. The drop in prefill speed on mobile device (unlike on server GPU) could be attributed to the lower compute capacity for parallel

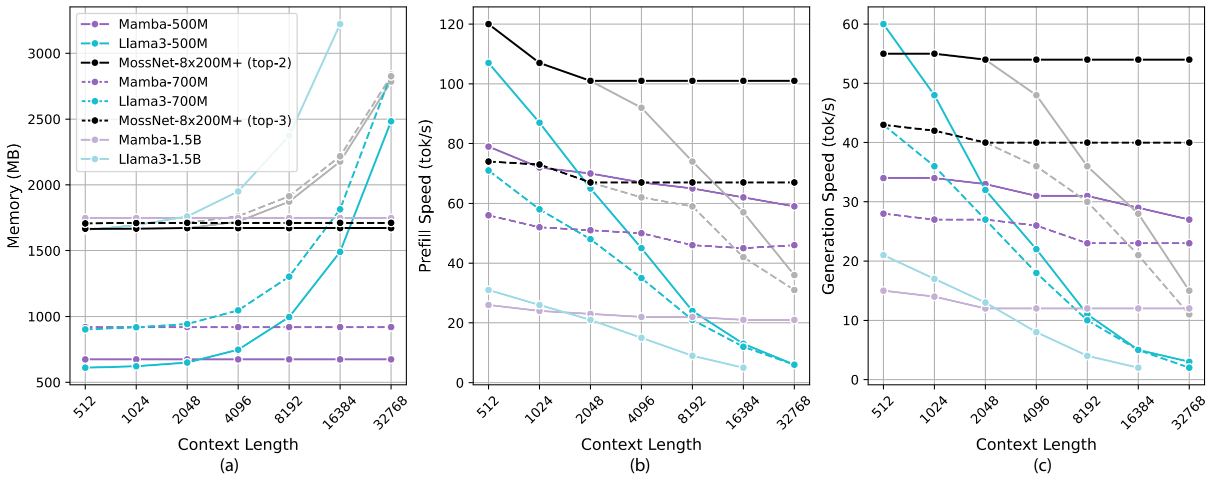

Figure 4: (a) Memory consumption, (b) prefill speed, and (c) generation speed with context length for MossNet-8x200M+ and baselines on Samsung Galaxy S24 Ultra (Q8 precision). Batch size set to 1. Gray line plots depict performance without SWA implemented. Llama3-1.5B results not plotted for 32K context due to out-of-memory error.

Table 3: Effect of architectural modifications to MossNet. Perplexity reported on Cosmopedia evaluation set.

Model	Tot. Param. (M)	Act. Param. (M)	PPL
MossNet-8x8M	19.7	9.9	13.1
w/o MHA	21.4	10.3	13.5
w/o MLP-MoE	19.1	9.8	13.4
w/ top-1, 8 experts	19.7	8.3	15.3
w/ top-4, 8 experts	19.7	13.2	12.6
w/ top-2, 4 experts	13.1	9.9	14.4
w/ top-2, 16 experts	32.7	9.9	12.0

processing. The stable performance and reduced resource use make MossNet especially suitable for on-device inference scenarios, where users often demand responsiveness and must operate under strict memory and compute constraints.

3.3 Architecture Modifications

We now test various modifications to the proposed MossNet architecture. We study the *relative* effect of removing MHA, MLP-MoE, and varying the number of total and activated experts. Table 3 summarizes the results. The proposed MossNet-8x8M achieves a PPL of 13.1 with 19.7M total parameters and 9.9M active parameters, demonstrating the effective use of MHA and MLP-MoE. Removing MHA increases parameters count and leads to modest performance drop (PPL = 13.5), while removing MLP-MoE yields fewer parameters but also worse perplexity (13.4).

Next, we test the effect of varying the number of total and activated experts. The table highlights a trade-off: activating fewer experts (e.g., top-1) can greatly hurt perplexity (up to 15.3), while activating more experts (e.g., top-4 with 8 experts) can reduce PPL to 12.6, at the cost of a higher active parameter count (resulting in higher memory and compute). Notably, employing 16 experts with top-2 activation gives the best perplexity (12.0) but increases the total parameter count to 32.7M, illustrating how scaling the MoE approach can yield lower perplexity with more overall model capacity.

4 Conclusion

In this paper, we introduced **MossNet**, a mixture-of-state-space-experts architecture designed to emulate a linear MHA within an SSM. By integrating MoE in both channel-mixing (MLP) and time-mixing (SSM) components, MossNet captures different temporal focus or scale of context, providing a richer representation than a single set of SSM parameters could. This is akin to the MHA mechanism in transformers. Our theoretical analysis shows that this approach indeed recovers a linearized form of MHA, and our empirical study on language modeling and downstream tasks demonstrates that MossNet outperforms both transformer-based and prior SSM/GRM-based baselines. Large-scale experiments further highlight its scalability and practical runtime benefits. We believe MossNet represents an important step toward fully harnessing recurrent models for language modeling at scale, opening up new directions for efficient, flexible, and high-performing LLM architectures.

5 Limitations

Despite the several advantages of the proposed MossNet architecture, there are several limitations. First, the integration of the MoE framework within both channel-mixing and time-mixing components of state-space models introduces considerable architectural complexity. This may present challenges for replication and broader adoption in the research community without specialized knowledge. MoEs do not effectively improve inference performance on server, when the input is a batch of user requests containing different tasks. Further, we evaluate MossNet on MLP tasks. We leave evaluation on more diverse downstream tasks such as multi-modal understanding, real-time applications, and specialized domains to future work. Finally, although MossNet shows promising results on mobile devices like the Samsung Galaxy S24 Ultra, performance across other hardware configurations, especially those with different architectures or constraints, may vary. Future work could explore adaptive optimizations tailored to specific hardware platforms.

References

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. 2023. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Lewis Tunstall, Agustín Piqueres, Andres Marafioti, Cyril Zakka, Leandro von Werra, and Thomas Wolf. 2024. Smollm2 - with great data, comes great performance.

Quentin Anthony, Yury Tokpanov, Paolo Glorioso, and Beren Millidge. 2024. BlackMamba: Mixture of experts for state-space models. *arXiv preprint arXiv:2402.01771*.

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, James Zou, Atri Rudra, and Christopher Re. 2024. Simple linear attention language models balance the recall-throughput tradeoff. In *Forty-first International Conference on Machine Learning*.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2023. Llemma: An open language model for mathematics. *arXiv preprint arXiv:2310.10631*.

Loubna Ben Allal, Anton Lozhkov, Guilherme Penedo, Thomas Wolf, and Leandro von Werra. 2024a. Cosmopedia.

Loubna Ben Allal, Anton Lozhkov, Guilherme Penedo, Thomas Wolf, and Leandro von Werra. 2024b. Smollm-corpus.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: a suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, pages 2397–2430.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7432–7439.

Sidney Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. 2022. GPT-NeoX-20B: An open-source autoregressive language model. In *Proceedings of BigScience Episode# 5–Workshop on Challenges & Perspectives in Creating Large Language Models*, pages 95–136.

Guy E. Blelloch. 1990. Prefix sums and their applications. Technical Report CMU-CS-90-190, School of Computer Science, Carnegie Mellon University.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. 2023. RT-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 2924–2936.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Tri Dao and Albert Gu. 2024. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. In *Forty-first International Conference on Machine Learning*.

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

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

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

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. 2024. A framework for few-shot language model evaluation.

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

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

Albert Gu, Karan Goel, and Christopher Re. 2022. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*.

Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. 2021. Combining recurrent, convolutional, and continuous-time models with linear state space layers. *Advances in neural information processing systems*, 34:572–585.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. In *Proceedings of the International Conference on Learning Representations*.

Samy Jelassi, David Brandfonbrener, Sham M Kakade, and Eran Malach. 2024. Repeat after me: Transformers are better than state space models at copying. *arXiv preprint arXiv:2402.01032*.

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

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.

Tobias Katsch. 2023. GateLoop: Fully data-controlled linear recurrence for sequence modeling. *arXiv preprint arXiv:2311.01927*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale reading comprehension dataset from examinations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 785–794.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, et al. 2024. Datacomp-lm: In search of the next generation of training sets for language models. *arXiv preprint arXiv:2406.11794*.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. 2023. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*.

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

Eric Martin and Chris Cundy. 2018. Parallelizing linear recurrent neural nets over sequence length. In *International Conference on Learning Representations*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? A new dataset for open book question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391.

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

OpenAI. 2024a. ChatGPT. `https://chatpgt.com`.

OpenAI. 2024b. Sora. `https://openai.com/index/sora`.

Badri Narayana Patro and Vijay Srinivas Agneeswaran. 2024. Mamba-360: Survey of state space models as transformer alternative for long sequence modelling: Methods, applications, and challenges. *arXiv preprint arXiv:2404.16112*.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Leon Derczynski, et al. 2023. Rwkv: Reinventing rnns for the transformer era. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14048–14077.

Maciej Pióro, Kamil Ciebiera, Krystian Król, Jan Ludziejewski, and Sebastian Jaszczur. 2024. MoE-Mamba: Efficient selective state space models with mixture of experts. *arXiv preprint arXiv:2401.04081*.

Zhen Qin, Songlin Yang, and Yiran Zhong. 2024. Hierarchically gated recurrent neural network for sequence modeling. *Advances in Neural Information Processing Systems*, 36.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 784–789.

Liliang Ren, Yang Liu, Yadong Lu, Yelong Shen, Chen Liang, and Weizhu Chen. 2024. Samba: Simple hybrid state space models for efficient unlimited context language modeling. *arXiv preprint arXiv:2406.07522*.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S. Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *Proceedings of the AAAI Spring Symposium Series*.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. 2023. Code Llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. WinoGrande: An adversarial Winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, et al. 2024. Dolma: An open corpus of three trillion tokens for language model pretraining research. *arXiv preprint arXiv:2402.00159*.

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

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2022. Efficient transformers: A survey. *ACM Computing Surveys*, 55(6):1–28.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*, 30:5998–6008.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800.

Xiaofeng Zhang, Yikang Shen, Zeyu Huang, Jie Zhou, Wenge Rong, and Zhang Xiong. 2022. Mixture of attention heads: Selecting attention heads per token. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4150–4162.

A Model Hyperparameters and Training Recipes

In this section, we provide details on the various model architecture hyperparameters and corresponding training recipes for the MossNet suite of models and baselines at different parameter scales.

Table 4 summarizes the design choices. Each row corresponds to a particular model variant, sorted by approximate total parameter count. The key columns indicate:

- Total and active parameters.
- Hidden and intermediate dimensions for the MLP layers.

Table 4: Key model architecture hyperparameters and training recipes for various baseline architectures (Pythia, Llama, Mistral, Mixtral, Griffin, Mamba, Mamba2, Zamba, and MoE-Mamba) alongside the proposed MossNet family of models. The table displays model sizes, dimensions, training tokens, context lengths, and learning rate schedules, among other relevant settings. α corresponds to the weight factor for load balancing loss. *Unlike MossNet-8x200M+ that was dynamically trained in top-2 and top-3 modes, smaller models were only trained in top-2 mode.

Model	**Tot. Param. (M)**	**Act. Param. (M)**	**Hidden Dim.**	**Intermediate Dim.**	**Num. Attn. Heads**	**Num. K/V Heads**	**Sliding Window**	**Num. Layers**	**Tie Embeddings**	**Train. Tokens (B)**	**Context Length**	**Max. LR**	**Schedule**	**Warmup**	**Final LR Ratio**	α
Pythia-9M	8.8	8.8	128	512	2	-	-	12	T	22	2048	2.0e-3	Cosine	3%	0%	-
Llama-8M	8.2	8.2	128	448	2	1	-	8	T	22	2048	2.0e-3	Cosine	3%	0%	-
Mistral-8M	8.2	8.2	128	448	2	1	256	8	T	22	2048	2.0e-3	Cosine	3%	0%	-
Mixtral-8x8M	17.8	9.6	128	448	2	1	256	8	T	22	2048	2.0e-3	Cosine	3%	0%	0.001
Griffin-9M	8.7	8.7	128	384	2	1	-	16	T	22	2048	1.0e-2	Cosine	3%	0%	-
Mamba-8M	8.3	8.3	128	448	-	-	-	16	T	22	2048	1.0e-2	Cosine	3%	0%	-
Mamba2-9M	8.6	8.6	128	256	-	-	-	16	T	22	2048	1.0e-2	Cosine	3%	0%	-
Zamba-8M	8.1	8.1	128	448	2	1	-	16	T	22	2048	1.0e-2	Cosine	3%	0%	-
MoE-Mamba-8x8M	16.8	9.7	128	384	-	-	-	16	T	22	2048	1.0e-2	Cosine	3%	0%	0.001
MossNet-8x8M*	19.7	9.9	128	384	2	1	-	16	T	22	2048	1.0e-2	Cosine	3%	0%	0.001
Pythia-22M	22.3	22.3	256	1,024	4	-	-	12	T	22	2048	1.0e-3	Cosine	3%	0%	-
Llama-20M	20	20	256	896	4	2	-	8	T	22	2048	1.0e-3	Cosine	3%	0%	-
Mistral-20M	20	20	256	896	4	2	256	8	T	22	2048	1.0e-3	Cosine	3%	0%	-
Mixtral-8x20M	58.5	25.5	256	896	4	2	256	8	T	22	2048	1.0e-3	Cosine	3%	0%	0.001
Griffin-22M	22	22	256	768	4	2	-	16	T	22	2048	5.0e-3	Cosine	3%	0%	-
Mamba-20M	19.9	19.9	256	896	-	-	-	16	T	22	2048	5.0e-3	Cosine	3%	0%	-
Mamba2-20M	20.4	20.4	256	512	-	-	-	16	T	22	2048	5.0e-3	Cosine	3%	0%	-
Zamba-20M	19.5	19.5	256	896	4	2	-	16	T	22	2048	5.0e-3	Cosine	3%	0%	-
MoE-Mamba-8x20M	54.1	25.8	256	768	-	-	-	16	T	22	2048	5.0e-3	Cosine	3%	0%	0.001
MossNet-8x20M*	63.9	26.1	256	768	4	2	-	16	T	22	2048	5.0e-3	Cosine	3%	0%	0.001
Pythia-64M	63.6	63.6	512	2048	8	-	-	12	T	22	2048	1.0e-3	Cosine	3%	0%	-
Llama-67M	66.7	66.7	512	1792	8	2	-	12	T	22	2048	1.0e-3	Cosine	3%	0%	-
Mistral-67M	66.7	66.7	512	1792	8	2	256	12	T	22	2048	1.0e-3	Cosine	3%	0%	-
Mixtral-8x67M	320.6	105.9	512	1792	8	2	256	12	T	22	2048	1.0e-3	Cosine	3%	0%	0.001
Griffin-61M	61.1	61.1	512	1792	8	2	-	16	T	22	2048	5.0e-3	Cosine	3%	0%	-
Mamba-66M	66.4	66.4	512	1792	-	-	-	24	T	22	2048	5.0e-3	Cosine	3%	0%	-
Mamba2-67M	67.2	67.2	512	1024	-	-	-	24	T	22	2048	5.0e-3	Cosine	3%	0%	-
Zamba-62M	62.1	62.1	512	1792	8	2	-	24	T	22	2048	5.0e-3	Cosine	3%	0%	-
MoE-Mamba-8x66M	272.6	102.8	512	1536	8	2	-	24	T	22	2048	5.0e-3	Cosine	3%	0%	0.001
MossNet-8x66M*	325.9	102.9	512	1536	8	2	-	24	T	22	2048	5.0e-3	Cosine	3%	0%	0.001
Pythia-330M	328.6	328.6	1024	4096	16	-	-	22	T	22	2048	3.0e-4	Cosine	3%	0%	-
Llama-350M	351.4	351.4	1024	3584	16	4	-	22	T	22	2048	3.0e-4	Cosine	3%	0%	-
Mistral-350M	351.4	351.4	1024	3584	16	4	512	22	T	22	2048	3.0e-4	Cosine	3%	0%	-
Griffin-330M	330.4	330.4	1024	3584	16	4	-	32	T	22	2048	1.5e-3	Cosine	3%	0%	-
Mamba-370M	371.5	371.5	1024	3584	-	-	-	48	T	22	2048	1.5e-3	Cosine	3%	0%	-
Mamba2-370M	369.9	369.9	1024	2048	-	-	-	48	T	22	2048	1.5e-3	Cosine	3%	0%	-
Zamba-330M	334.1	334.1	1024	3584	16	4	-	48	T	22	2048	1.5e-3	Cosine	3%	0%	-
MossNet-8x200M+	1554.5	477/657	1024	3072	16	4	2048	30	F	2800	4096	2.0e-4	WSD	1%	10%	0.001

- Total number of attention heads and K/V heads (for grouped-query attention; Ainslie et al. 2023).
- Sliding window size, if applied.
- Number of layers.
- Tie embeddings, i.e., whether input and output embeddings are tied.
- Total number of training tokens.
- Context length used for training.
- Other training hyperparameters, including α, i.e., the weight factor used for load balancing loss in MoE architectures.

We group models by approximate size categories, illustrating how scaling up parameters impacts the choice of dimensionality and training regimes.

Note that we train MossNet-8x200M+ in a dynamic setting. We train the model in top-3 mode

Table 5: Composition of pre-training data for MossNet-8x200M+.

Dataset	Num. Tok. (B)	Train Tok. (B)	Samp. Wgt.
DCLM-baseline	4000	2520	0.90
Starcoder	250	168	0.06
Proof-Pile-2	55	28	0.01
peS2o	47	28	0.01
Cosmopedia-2	28	56	0.02
Total		2800	1.00

for 900 steps and in top-2 mode for 100 steps and repeat the cycle. All models are trained on the Cosmopedia dataset (fair training setting), except MossNet-8x200M+ that we train on a custom pre-training dataset.

B Custom Pre-training Dataset

Table 5 shows the mixture of open datasets that form the custom pre-training data mix for MossNet-8x200M+. We combine DCLM-baseline (Li et al., 2024), Starcoder (Li et al., 2023), Proof-Pile-2 (Azerbayev et al., 2023), peS2o (Soldaini et al., 2024), and Cosmopedia-2 (Ben Allal et al., 2024b) with different sampling weights.

C Additional Results

In this section, we present additional results.

C.1 Commonsense Performance of Fairly-trained Models

Fig. 2 summarizes the commonsense performance of MossNet and baseline models. Table 6 presents the detailed results. Again, MossNet outperforms baseline architectures at different active parameter scales. We scale parameter sizes up to 100M and leave experiments on larger models to future work.

C.2 Speed and Memory Results

Figs. 3 and 4 summarize the speed and memory performance of MossNet-8x200M+ and baseline models at different active parameters scales. We present the detailed results for GPU profiling in Tables 7, 8, and 9 for memory consumption, prefill speed, and generation speed, respectively. We also present the detailed results for mobile profiling in Tables 10, 11, and 12.

C.3 Long-context Performance

Table 13 presents the long context performance of MossNet-8x200M+ and various baselines. We observe that architectures using SSM and/or SWA backbones do not lose perplexity as the context size is increased. This confines with the observations of Ren et al. (2024).

C.4 Choosing k

Table 3 varies the number of routed experts (k) while keeping all other hyper-parameters fixed. The main observations are:

- **Large first step, then saturation.** With eight experts, increasing k from 1 to 2 reduces perplexity from 15.3 to 13.1 (-2.2), whereas a further increase to $k = 4$ only improves perplexity to 12.6 (-0.5) while significantly increasing active parameter count ($+3.3$).
- **Pool size matters.** Holding $k = 2$ and shrinking the pool from 8 to 4 experts worsens perplexity (13.1 to 14.4). Conversely, expanding the pool to 16 experts (still routing $k = 2$) attains the best perplexity (12.0) *without* increasing *active* parameters, although the *total* model size grows, resulting in a larger disk size.

We propose the following practical rule-of-thumb:

$$k = \min\Big(2, \ \lfloor N_{\text{experts}}/4 \rfloor\Big)$$

This caps compute at $\leq 2\times$ the dense baseline, preserving MossNet's on-device speed advantage. It also maintains high router entropy; choose $k = 1$ only when $N_{\text{experts}} < 8$. Finally, it secures $\geq 80\%$ of the achievable perplexity gain while avoiding the potential latency hit for $k \geq 4$.

C.5 Computational complexity of MoE blocks (and MossNet)

MossNet replaces the dense channel-/time-mixing layers in Mamba with *standard* top-k MoE blocks. Because only k experts are executed per token, its *time* complexity is $\Theta(L\,k\,d\,d_{\text{ff}})$ and its *activation memory* is $\Theta(k\,d)$—identical to other MoE architectures for the same k. Thus MossNet offers the usual "capacity without extra compute" benefit of MoE while preserving the linear-time, constant-cache profile of its dense counterpart.

Table 6: Zero-shot performance of MossNet and fairly-trained baselines on commonsense tasks.

Model	ARC-c	ARC-e	COPA	HellaSwag	OBQA	PIQA	WinoGrande	Average
Pythia-9M	18.3	32.6	**57.0**	26.5	**14.6**	56.2	50.8	36.6
Llama-8M	18.8	34.0	55.0	26.5	11.6	56.6	**52.8**	36.5
Mistral-8M	19.2	33.5	56.0	26.4	11.2	56.4	50.1	36.1
Mixtral-8x8M	19.5	**35.0**	53.0	27.1	13.0	56.8	50.3	36.4
Griffin-9M	19.1	32.2	48.0	26.0	14.6	56.2	50.8	35.3
Mamba-8M	19.3	34.0	51.0	26.4	13.2	58.0	51.5	36.2
Mamba2-9M	**19.8**	34.8	52.0	26.5	12.2	57.3	51.0	36.1
Zamba-8M	18.1	33.2	53.0	26.2	14.0	57.4	50.9	36.1
MoE-Mamba-8x8M	18.5	34.7	53.0	25.8	15.2	55.0	47.4	35.6
MossNet-8x8M	18.6	34.5	55.0	**27.7**	14.0	**59.6**	49.8	**37.1**
Pythia-22M	19.8	37.8	63.0	27.1	13.4	56.6	50.6	38.3
Llama-20M	17.9	35.4	56.0	26.9	11.2	58.2	52.1	36.8
Mistral-20M	20.0	35.9	54.0	26.8	13.8	58.8	50.4	37.1
Mixtral-8x20M	19.5	36.2	54.0	28.2	14.4	59.1	48.9	37.2
Griffin-22M	20.2	29.5	**57.0**	26.5	14.4	56.4	49.7	36.2
Mamba-20M	18.1	**39.0**	55.0	27.0	15.2	59.5	51.4	37.9
Mamba2-20M	18.3	36.2	55.0	27.3	14.4	57.9	50.8	37.1
Zamba-20M	20.3	38.5	55.0	27.2	14.4	58.6	**52.3**	38.0
MoE-Mamba-8x20M	19.8	38.1	53.0	27.5	16.8	58.8	48.2	37.5
MossNet-8x20M	**20.7**	38.8	**57.0**	**28.8**	**15.6**	**61.2**	50.5	**38.9**
Pythia-64M	20.4	41.5	51.0	28.3	15.6	61.8	**51.5**	38.6
Llama-67M	21.1	41.8	56.0	28.8	17.2	60.3	51.3	39.5
Mistral-67M	20.6	40.7	57.0	28.2	17.4	61.8	49.7	39.4
Mixtral-8x67M	22.4	**46.2**	**58.0**	31.3	17.0	63.2	50.5	41.2
Griffin-61M	21.9	30.2	56.0	26.8	16.8	57.7	51.0	37.2
Mamba-66M	21.2	44.4	48.0	29.0	17.4	63.0	50.8	39.1
Mamba2-67M	21.2	44.2	56.0	29.2	16.8	62.3	45.6	39.3
Zamba-62M	21.3	40.8	54.0	28.8	15.6	61.1	49.6	38.7
MoE-Mamba-8x66M	22.4	45.2	56.0	30.4	18.6	61.8	48.7	40.4
MossNet-8x66M	**22.6**	45.1	**58.0**	**31.5**	**19.4**	**65.5**	51.0	**41.9**
Pythia-330M	21.8	45.2	55.0	29.5	16.6	62.4	51.1	40.2
Llama-350M	23.2	45.2	58.0	30.6	16.2	63.2	50.1	40.9
Mistral-350M	20.9	46.1	55.0	30.4	19.2	61.6	50.1	40.5
Griffin-330M	20.3	36.7	58.0	29.4	19.4	61.0	49.1	39.1
Mamba-370M	23.9	**55.8**	59.0	33.0	20.2	**66.5**	51.6	**44.3**
Mamba2-370M	**25.4**	50.4	**60.0**	**33.1**	**21.2**	65.1	51.9	43.9
Zamba-330M	24.2	48.9	56.0	32.6	19.4	64.6	**53.3**	42.7

Table 7: Memory (GB) of various models on A100-80 GPU (F16 precision, FlashAttention 2) across varying prompt lengths. Batch sizes are denoted as **1** and **4**.

	Model	Batch Size 1							Batch Size 4						
		512	1024	2048	4096	8192	16384	32768	512	1024	2048	4096	8192	16384	32768
~500M	Mamba-500M	2.5	2.5	2.6	2.8	3.4	4.5	8.9	2.7	2.8	3.2	4.1	6.0	10.6	32.8
	Llama3-500M	3.6	3.8	4.2	4.9	6.4	9.5	15.6	4.2	4.9	6.4	9.5	15.6	28.0	52.5
	MossNet-8x200M+ (top-2)	4.6	4.7	4.8	5.0	5.5	6.5	8.4	4.8	5.1	5.5	6.5	8.4	12.2	19.9
~700M	Mamba-700M	3.1	3.2	3.3	3.6	4.3	6.1	10.0	3.3	3.6	4.2	5.6	8.4	15.6	33.5
	Llama3-700M	4.3	4.5	4.8	5.6	7.2	10.6	17.1	4.8	5.6	7.2	10.6	17.1	29.6	56.6
	MossNet-8x200M+ (top-3)	4.6	4.7	4.8	5.1	5.6	6.7	8.8	4.8	5.1	5.6	6.7	8.8	13.0	23.9
~1.5B	Mamba-1.5B	4.6	4.6	4.8	5.2	6.1	8.3	12.6	4.9	5.2	6.0	7.7	11.1	18.7	37.2
	Rene-1.3B	4.6	4.6	4.6	4.6	5.1	6.4	9.2	4.6	4.8	5.3	6.4	9.2	13.8	23.3
	Hymba-1.5B	5.0	5.3	5.9	6.9	8.4	12.0	19.0	6.1	7.2	8.5	12.2	19.2	33.3	61.5
	Llama3-1.5B	4.3	4.5	4.8	5.6	7.2	10.6	17.1	4.8	5.6	7.2	10.6	17.1	29.6	56.6

Table 8: Prefill speed ($\times 10^3$ tok/s) of various models on A100-80 GPU (F16 precision, FlashAttention 2) across varying prompt lengths. Batch sizes are denoted as **1** and **4**.

	Model	**Batch Size 1**							**Batch Size 4**						
		512	**1024**	**2048**	**4096**	**8192**	**16384**	**32768**	**512**	**1024**	**2048**	**4096**	**8192**	**16384**	**32768**
~500M	Mamba-500M	8.6	17.8	29.3	35.6	41.0	44.4	46.5	28.0	34.5	38.8	43.4	46.7	47.6	46.9
	Llama3-500M	14.3	27.8	48.2	65.7	76.1	88.7	94.6	47.2	74.3	89.8	93.0	95.8	96.8	96.4
	MossNet-8x200M+ (top-2)	10.8	21.6	40.4	48.6	52.7	64.0	75.8	40.6	49.5	54.0	66.1	76.9	84.0	87.1
~700M	Mamba-700M	11.8	23.6	29.3	33.9	36.9	40.0	40.8	28.1	31.8	36.2	39.2	41.0	41.0	40.1
	Llama3-700M	18.7	34.6	55.2	67.7	72.6	76.4	78.3	56.8	71.8	76.5	78.0	78.6	78.9	78.4
	MossNet-8x200M+ (top-3)	11.3	22.1	33.4	38.5	44.8	57.7	65.4	33.9	39.2	41.8	59.0	67.9	71.7	74.3
~1.5B	Mamba-1.5B	9.9	15.3	18.5	20.2	23.2	24.8	25.2	17.9	19.7	22.2	23.4	24.3	24.2	23.8
	Rene-1.3B	7.7	15.4	31.0	42.0	44.2	46.6	47.0	31.0	50.1	48.3	48.7	48.1	47.6	47.4
	Hymba-1.5B	6.9	13.5	16.2	19.1	21.9	22.6	22.3	15.4	17.9	19.9	22.6	23.7	23.4	22.4
	Llama3-1.5B	10.8	18.8	26.8	30.6	35.1	36.5	37.1	29.8	34.7	37.5	37.1	37.4	37.2	36.8

Table 9: Generation speed (tok/s) of various models on A100-80 GPU (F16 precision, FlashAttention 2) across varying prompt lengths. Batch sizes are denoted as **1** and **4**.

	Model	**Batch Size 1**							**Batch Size 4**						
		512	**1024**	**2048**	**4096**	**8192**	**16384**	**32768**	**512**	**1024**	**2048**	**4096**	**8192**	**16384**	**32768**
~500M	Mamba-500M	17.9	18.3	17.7	18.0	18.1	17.8	17.9	68.3	70.5	71.8	69.8	70.1	70.4	70.6
	Llama3-500M	28.8	29.2	27.7	28.2	27.6	26.2	22.7	110.4	113.5	108.3	101.8	92.2	76.0	55.9
	MossNet-8x200M+ (top-2)	27.5	28.6	27.0	28.1	27.9	27.9	27.7	91.2	92.5	91.1	90.3	89.5	86.5	81.0
~700M	Mamba-700M	24.5	24.7	25.0	24.8	24.7	24.6	25.2	98.0	95.1	95.7	97.4	96.6	94.56	99.0
	Llama3-700M	37.1	40.1	39.0	38.9	37.4	33.8	29.6	158.0	151.0	144.0	133.5	113.5	87.6	61.0
	MossNet-8x200M+ (top-3)	28.7	28.6	28.1	28.3	28.0	27.4	26.9	90.2	91.7	92.4	91.6	87.1	85.2	53.0
~1.5B	Mamba-1.5B	20.7	21.1	21.0	21.1	20.6	20.8	20.6	81.3	82.0	82.4	80.8	81.9	82.0	81.7
	Rene-1.3B	27.3	27.0	27.3	27.0	27.0	26.6	26.7	107.6	105.5	106.5	106.9	107.6	107.4	108.8
	Hymba-1.5B	14.9	14.9	14.6	14.9	14.5	14.6	14.6	58.4	56.3	57.0	57.2	56.3	56.6	56.4
	Llama3-1.5B	22.2	22.4	21.2	21.4	20.9	18.3	15.0	88.3	85.7	80.5	69.7	58.5	43.3	28.2

Table 10: Memory (MB) of various models on S24 Ultra (Q8 precision) across varying prompt lengths and active parameters. Batch size set to 1. MossNet on-device results reported without SWA implemented.

	Model	**512**	**1024**	**2048**	**4096**	**8192**	**16384**	**32768**
~500M	Mamba-500M	673.7	673.7	673.7	673.7	673.7	673.7	673.7
	Llama3-500M	610.6	621.6	649.6	747.4	995.4	1491.4	2483.4
	MossNet-8x200M+ (top-2)	1666.9	1667.7	1670.7	1720.7	1872.7	2176.7	2784.7
~700M	Mamba-700M	919.3	919.3	919.3	919.3	919.3	919.3	919.3
	Llama3-700M	901.6	917.6	942.6	1046.4	1302.4	1814.4	2838.4
	MossNet-8x200M+ (top-3)	1707.0	1709.9	1711.9	1761.8	1913.8	2217.8	2825.8
~1.5B	Mamba-1.5B	1748.0	1748.0	1748.0	1748.0	1748.0	1748.0	1748.0
	Llama3-1.5B	1657.2	1694.2	1760.2	1950.0	2374.0	3222.0	OOM

Table 11: Memory (MB) of various models on S24 Ultra (Q8 precision) across varying prompt lengths and active parameters. Batch size set to 1. MossNet on-device results reported without SWA implemented.

	Model	**512**	**1024**	**2048**	**4096**	**8192**	**16384**	**32768**
~500M	Mamba-500M	673.7	673.7	673.7	673.7	673.7	673.7	673.7
	Llama3-500M	610.6	621.6	649.6	747.4	995.4	1491.4	2483.4
	MossNet-8x200M+ (top-2)	1666.9	1667.7	1670.7	1720.7	1872.7	2176.7	2784.7
~700M	Mamba-700M	919.3	919.3	919.3	919.3	919.3	919.3	919.3
	Llama3-700M	901.6	917.6	942.6	1046.4	1302.4	1814.4	2838.4
	MossNet-8x200M+ (top-3)	1707.0	1709.9	1711.9	1761.8	1913.8	2217.8	2825.8
~1.5B	Mamba-1.5B	1748.0	1748.0	1748.0	1748.0	1748.0	1748.0	1748.0
	Llama3-1.5B	1657.2	1694.2	1760.2	1950.0	2374.0	3222.0	OOM

Table 12: Prefill speed (tok/s) of various models on S24 Ultra (Q8 precision) across varying prompt lengths and active parameters. Batch size set to 1. MossNet on-device results reported without SWA implemented.

	Model	512	1024	2048	4096	8192	16384	32768
~500M	Mamba-500M	79	72	70	67	65	62	59
	Llama3-500M	107	87	65	45	24	13	6
	MossNet-8x200M+ (top-2)	120	107	101	92	74	57	36
~700M	Mamba-700M	56	52	51	50	46	45	46
	Llama3-700M	71	58	48	35	21	12	6
	MossNet-8x200M+ (top-3)	74	73	67	62	59	42	31
~1.5B	Mamba-1.5B	26	24	23	22	22	21	21
	Llama3-1.5B	31	26	21	15	9	5	OOM

Table 13: Long context performance of MossNet and various baselines. Training tokens and whether SWA is implemented for corresponding models are also provided. *Perplexity should not be compared directly for different models as they could be using different tokenizers; trend with context size should be observed instead.

Model	Train Tok.	SWA	PPL*			
			2048	4096	8192	16384
Phi1.5	0.15T	F	11.8	70.6	441.1	OOM
SmolLM2-1.7B	11T	F	9.8	83.6	590.8	OOM
Mamba-1.4B	0.3T	-	6.9	6.7	7.2	OOM
MossNet-8x200M+	2.8T	T	8.6	8.2	8.0	8.7