Sparsifying Mamba

An Wang[1], Ruobing Xie[1,*], Shuaipeng Li[1], Xingwu Sun[1,2], Zhanhui Kang[1]

[1]Tencent Hunyuan
[2]University of Macau

Abstract

The Transformer architecture has long dominated the development of large language models, but its quadratic complexity in sequence length presents scalability challenges. Recent advances in State Space Models, particularly Mamba series, offer a promising alternative with linear-time inference and competitive performance. While scaling model capacity via sparsification, exemplified by Mixture-of-Experts, has proven effective in reducing computation while expanding knowledge capacity, the integration of sparsification with Mamba remains largely unexplored. Existing attempts typically apply naive block-level stacking, failing to leverage Mamba's internal structure for fine-grained sparsification. In this work, we mainly explore how to sparsify the parameters inside Mamba. We found that the effects of using sparsification strategies on parameters related to various mechanisms inside mamba are significantly different. Our proposed MambaMoZ framework introduces a flexible and effective sparsification mechanism inside Mamba, which can independently achieve parameter scalability and has stronger performance.

1 Introduction

The evolution of language models has been profoundly shaped by the Transformer architecture (Vaswani et al., 2017), which introduced self-attention mechanisms to capture long-range dependencies in sequential data. While Transformers have dominated large-scale language modeling tasks, their quadratic computational complexity and memory demands pose significant challenges for scaling to ever-growing datasets. Recent advancements in sequence modeling have introduced State Space Models as a promising alternative, with Mamba (Gu and Dao, 2023; Dao and Gu, 2024) emerging as a standout architecture due to its selective state mechanism and linear-time inference efficiency. By dynamically adjusting parameters based on input context, Mamba achieves performance that is competitive with attention-based models while dramatically reducing computational overhead.

As modern AI systems increasingly rely on scaling model capacity to harness massive datasets, traditional dense architectures face prohibitive training costs and hardware limitations. A proven strategy to address this is Mixture-of-Experts (MoE) sparsification (Lepikhin et al., 2020; Fedus et al., 2022), where only subsets of specialized subnetworks ("experts") activate per input. This paradigm decouples model size from computation, enabling knowledge storage across multiple experts without proportional increases in FLOPs. Recently, sparse architectures have been verified to preserve efficiency while benefiting from scaling laws (Ludziejewski et al., 2024). However, such a phenomenon remains underexplored in SSM-based models like Mamba. Could Mamba's core innovations be combined with sparsification to unlock new Pareto frontiers in accuracy-efficiency tradeoffs?

Pioneer efforts to integrate Mamba with MoE have focused on naive block-level stacking, such as interleaving Mamba layers with sparse feed-forward networks (Pióro et al., 2024). However, these approaches inherit two critical limitations: (i) They simply treat Mamba as a monolithic operator rather than exploring internal structural opportunities for sparsification, potentially overlooking component-specific optimization pathways; (ii) The ad hoc combination lacks systematic analysis of how sparsification interacts with Mamba's unique mechanism, risking suboptimal parameter utilization. Consequently, existing designs do not consider the effective integration of Mamba and sparsification techniques inside Mamba.

In this work, we conduct a systematic exploration of ***various sparsification strategies*** meticu-

*Corresponding Author.

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 17661–17667
November 4-9, 2025

lously designed ***inside Mamba***. Notably, the majority of Mamba's parameters are densely concentrated within its linear layers. Accordingly, we prioritize sparsification efforts on these parameter-intensive components across various functional mechanisms within the model. To distinguish their roles, we categorize these layers into two groups: those involved in context-information fusion and those not. Our experiments reveal that sparsifying layers related to the context-information fusion mechanism yields no significant performance improvement. In contrast, sparsifying the non-fusion layers—specifically the ***z branch in Mamba***—leads to notable performance gains. These findings not only deepen our understanding of sparsification strategies for Mamba but also provide a foundation for developing more efficient and scalable language models. The proposed sparsified Mamba variants emerge as a highly promising alternative to the original Mamba block, paving the way for further advancements in LLMs.

2 Methods

2.1 Basic Structure of Mamba

Mamba (Gu and Dao, 2023) introduces a novel approach to sequence modeling through SSMs. Its successor, Mamba2 (Dao and Gu, 2024), further enhances this framework with tensorized attention mechanisms. Below, we detail the computational formulations of Mamba2 architectures.

The Mamba2 (Dao and Gu, 2024) architecture builds upon its predecessor by incorporating a tensorized reparameterization technique that merges the strengths of state-space models with attention mechanisms. The core computations of Mamba2 are described as follows:

$$\begin{aligned} \mathbf{B} &= \delta\left(\mathbf{x}_k \mathcal{W}_{\mathbf{B}}\right) \\ \mathbf{C} &= \delta\left(\mathbf{x}_k \mathcal{W}_{\mathbf{C}}\right) \\ \mathbf{X} &= \sigma\left(\mathbf{x}_k \mathcal{W}_{\Delta} + b_{\Delta}\right) \mathbf{x}_k \\ \mathbf{y} &= \left(\mathbf{A}^{\times} \circ \mathbf{C}\mathbf{B}^T\right) \mathbf{X} \\ \mathbf{o}_k &= \mathbf{y}_k \times \delta(\mathbf{x}_k \mathcal{W}_{\mathbf{z}}) \end{aligned} \tag{1}$$

Here, σ denotes the softplus (Zheng et al., 2015) activation function, and δ represents the SiLU (Sigmoid Linear Unit) function. The architecture features input-dependent parameterization through Δ_k, which modulates both the discretization rate and the effective state transition. In this formulation, the Hadamard product ($\circ$) between the learnable tensor $\mathbf{A}^{\times}$ and the outer product $\mathbf{C}\mathbf{B}^T$ allows for a structured interaction between the different components. This operation enables Mamba2 to capture both global and local relationships within the data, enhancing its expressive power. Because Mamba2 outperforms the original Mamba in both performance and speed, our new methods are proposed based on Mamba2 architecture.

2.2 Sparsifying Linear Layers inside Mamba

In the Mamba2 architecture, we explore the sparsification of all linear layers in the model, specifically replacing a single parameter matrix with multiple parameter matrices and a routing mechanism as shown in Figure 1. This design aims to reduce the computational complexity of the model while maintaining its expressive power. By introducing a sparse routing mechanism, we allow the model to selectively activate different transformations based on the input, which can lead to more efficient computations and a better trade-off between performance and resource utilization.

The sparsification of linear layers is applied to the weights $\mathcal{W}_B$, $\mathcal{W}_C$, $\mathcal{W}_\Delta$, and $\mathcal{W}_z$ in the Mamba block, where traditional dense layers are often the computational bottleneck. By applying sparsification, we aim to capture essential features with fewer active parameters, while the routing mechanism ensures that only the most relevant transformations are applied at each step, leading to improved scalability and efficiency in training and inference.

2.3 Sparsifying Z Branch inside Mamba

In this subsection, we focus on the sparsification of the $\mathcal{W}_\mathbf{z}$ matrix in the computation $\mathbf{o}_k = \mathbf{y}_k \times \delta(\mathbf{x}_k \mathcal{W}_\mathbf{z})$ in the Mamba2 model. The matrix $\mathcal{W}\mathbf{z}$ represents a linear transformation that affects the output $\mathbf{o}_k$, and sparsifying it has the potential to reduce the number of parameters involved in this critical computation.

The sparsification of $\mathcal{W}_\mathbf{z}$ follows a similar logic to the sparsification of linear layers described in the previous subsection, but with a focus on transformations that are not dependent on positional information. In the Mamba model, there are two types of linear transformations: those that operate on position-dependent data and those that are position-agnostic. We hypothesize that sparsifying the position-agnostic transformations yields greater benefits in terms of both efficiency and effectiveness, as these transformations do not require intricate contextual information tied to specific positions in the sequence.

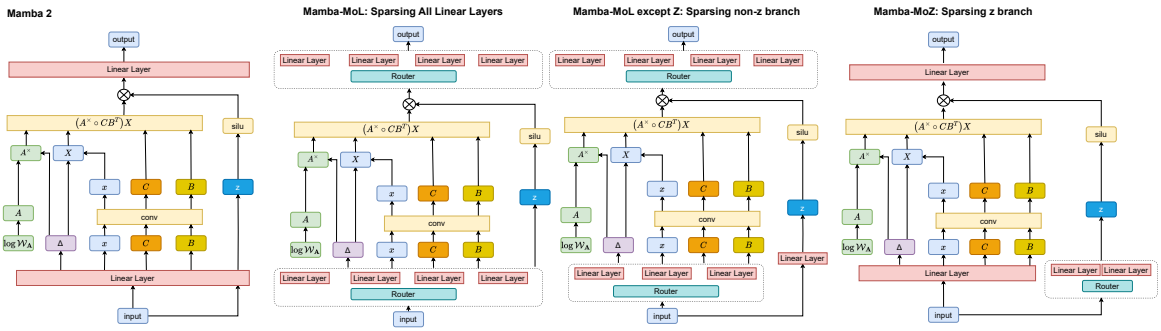

Figure 1: Overview of the original structure and different intra-sparsification strategies of Mamba.

This approach aligns with current trends in the use of MoE (Mixture of Experts) models (Dai et al., 2024; Jiang et al., 2024; Wang et al., 2024), where attention layers remain dense, but FFN layers are sparsified to optimize the model's capacity and efficiency. By adopting this strategy, Mamba2 can maintain its expressive power while reducing the overall complexity of the model.

3 Experiment

To evaluate the effectiveness of sparsifying different components in Mamba, we conduct experiments on three benchmark tasks (we use accuracy as the metric, with a 0-shot evaluation setting): ARC (Clark et al., 2018), HellaSwag (Zellers et al., 2019), SIQA (Sap et al., 2019). All models are trained at the 650M activated parameter and 1.3B total parameter scale (for sparsified mamba) under comparable conditions. Our experiments are organized to address two key questions: (1) Which sparsification strategy best improves accuracy and efficiency trade-offs in Mamba? (2) How do hybrid architectures, combining the original Mamba blocks and the MoE blocks, perform in comparison to fully sparse Mamba variants?

3.1 Model Setting

For all Mamba and its variant models, we configure 48 Mamba blocks, where the hidden dimension of each Mamba block is set to 1024. In the mamba layer, we employ 16 mamba2 heads, each with a dimension of 64. To ensure the total parameter count remains consistent across different models, our Mamba-MoL model utilizes 4 experts in total; the Mamba-MoL, except Z, employs 5 experts; the Mamba-MoZ model uses 7 experts. During both training and inference, only one expert is activated.

3.2 Training Settings

All models are trained on 8 NVIDIA H800 GPUs, each equipped with 80 GB of memory. We employ the AdamW optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a weight decay coefficient of 1×10^{-5}. The learning rate is linearly warmed up from 0 to 1×10^{-4} over the first 1000 steps, after which it remains constant. Each model is trained with a context length of 4096 tokens and a global batch size of 1024. To ensure reproducibility, all experiments use a fixed random seed of 12345. All model implementations and training procedures are developed using the Megatron-LM framework (Shoeybi et al., 2019). The training corpus consists of a mix of web data, code data, and mathematical data.

3.3 Competitors

To assess the effectiveness of our proposed sparsification strategies, we compare several variants of Mamba, summarized below:

- **Mamba2 (Baseline)** (Dao and Gu, 2024): The standard dense Mamba model serving as the performance reference for all sparsified variants.
- **MoE-Mamba** (Pióro et al., 2024): A hybrid model alternating dense Mamba blocks with sparse expert-based modules, mimicking stacked sparsification without modifying internal Mamba structure.
- **Mamba-MoL**: Applies Mixture-of-Experts (MoE) sparsification to all linear transformation layers in Mamba2, including $\mathcal{W}\mathbf{z}$.
- **Mamba-MoL except Z**: Applies sparsification to all linear layers *except* for the $\mathcal{W}\mathbf{z}$ branch, allowing us to isolate its contribution.
- **Mamba-MoZ**: Applies sparsification *only* to the z-branch linear transformation, preserving the rest of the Mamba model in dense form.

Model Variant	Activated Params	Total Params	Training Tokens	Scores			
				ARC	Hellaswag	SIQA	AVG
Mamba	650M	650M	160B	27.42	37.37	33.93	32.91
Mamba-MoL	650M	1.3B	160B	25.75	38.87	33.57	32.73
Mamba-MoL except Z	650M	1.3B	160B	26.75	38.88	34.85	33.49
Mamba-MoZ	650M	1.3B	160B	**32.11**	**39.21**	**38.23**	**36.52**

Table 1: Model Performance Comparison. AVG means average performance across three benchmarks.

These settings allow us to explore both fine-grained and block-level sparsification. In particular, Mamba-MoZ isolates the effect of sparsifying position-agnostic components and serves as a testbed for evaluating targeted internal sparsification strategies.

3.4 Results and Model Analyses

We analyze the impact of different sparsification strategies on downstream performance and compare their efficiency and effectiveness under consistent training conditions. All models have roughly comparable activation parameters, computational cost, and training speed. Table 1 summarizes the accuracy on ARC, HellaSwag, and SIQA at 160B training tokens. The key findings are:

(a) Sparsification brings benefit. Targeted sparsification consistently improves model efficiency without catastrophic accuracy losses, demonstrating its potential to reduce computational overhead while preserving— or even enhancing—task performance. However, the gains are highly dependent on the choice of sparsification strategy, as indiscriminate sparsity can degrade performance if improperly applied.

(b) Full Sparsification is Suboptimal. Applying sparsification across all linear layers (Mamba-MoL) may lead to degraded performance across part of tasks. This suggests that Mamba blocks contain critical, non-redundant computations—particularly those governing sequential dynamics—that are disrupted by indiscriminate sparsity. The adverse effects highlight the need for selective sparsity to balance efficiency and expressiveness.

(c) Selective Sparsification of the z-Branch is better. Remarkably, sparsifying only the z-branch (Mamba-MoZ) outperforms all other configurations, achieving higher accuracy with equivalent or lower FLOPs. This underscores the z-branch's unique role in position-agnostic output modulation: expert selection within this branch enhances model expressiveness without compromising sequential dependencies. Notably, Mamba-MoZ significantly outperforms Mamba-MoL (excluding the z-branch), demonstrating that isolating sparsity to context-independent parameters (e.g., the z-branch) is a more effective sparsification strategy.

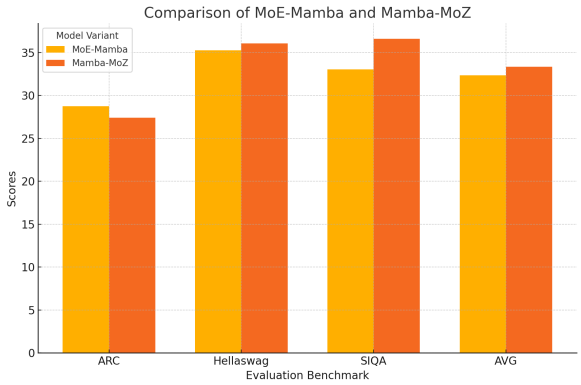

Figure 2: Comparison between MoE-Mamba with Mamba-Moz.

These results collectively demonstrate that sparsification in Mamba is not just about reducing FLOPs—it must be applied strategically. By isolating sparsification to the most effective components (like the z-branch), models can achieve higher accuracy with less computation, pushing the Pareto frontier in accuracy-efficiency trade-offs.

Sparsified Mamba vs. MoE-Mamba. We also compared the hybrid structure that combines Mamba and Moe with the pure sparse Mamba as shown in Figure 2. We found that the effect of pure sparse Mamba is comparable to or even better than the hybrid while ensuring the simplicity of the model structure. This proves the effectiveness of the Mamba-MoZ structure we proposed.

3.5 Loss Comparision

We compared the loss curves of models using different sparsification strategies in Figure 3, including the original mamba and its variants, after applying the sparsification strategy. We found that the loss

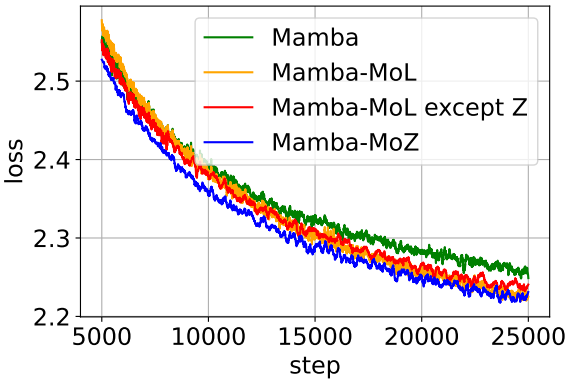

Figure 3: Loss comparison between models using different sparsification strategies.

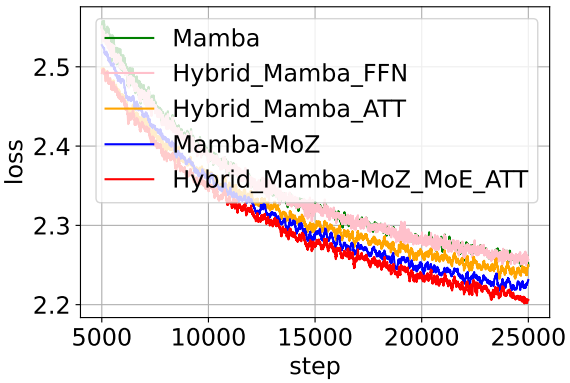

Figure 4: Loss comparison between different hybrid models. Hybrid_Mamba_FFN is to replace half of the Mamba model layers with FFN layers of the same size. Hybrid_Mamba_ATT is to replace 4 layers of the Mamba model with attention layers of the same size. Hybrid_Mamba-MoZ_MoE_ATT replaces half of the model layers with MoE layers of the same size, and then replaces the remaining 4 Mamba sparse layers with Attention layers of the same activation size.

curve of the original mamba model was the highest, and after applying the sparsification strategy, the loss curve dropped significantly. Among them, the loss curve using MoZ was the lowest. This proves the superiority of our proposed strategy.

Recent work (Lieber et al., 2024; Blakeman et al., 2025) has demonstrated the benefits of integrating Mamba modules with Transformer components. To investigate this, we replaced selected layers of a pure Mamba model with MLP and attention layers. As shown in Figure 4, while substituting MLP layers had a negligible impact on loss, introducing a small number of attention layers led to a notable reduction. This highlights the role of attention in enhancing the expressive capacity of Mamba models. Extending this approach, we incorporated MoE and attention layers into the Mamba-MoZ architecture. Under the same total parameter budget but with fewer activated parameters, the model achieved further loss reductions, confirming both the effectiveness of Mamba-MoZ and its compatibility with other modules.

4 Related Work

Transformers (Vaswani et al., 2017) dominate sequence modeling but face quadratic complexity challenges. State Space Models (SSMs) like Mamba (Gu and Dao, 2023) address this with linear-time inference, though prior work (Pióro et al., 2024) mainly applies MoE at the block level without internal sparsification. Sparse architectures like Mixture-of-Experts (Lepikhin et al., 2020; Fedus et al., 2022) improve efficiency but remain largely unexplored in SSM contexts. While network pruning (Liu et al.) targets dense models, they lack mechanisms for SSM's temporal dynamics. Our work pioneers component-aware sparsification within Mamba, particularly targeting its linear layers, which contrasts with prior uniform or block-level approaches. This targeted strategy optimizes both efficiency and performance by leveraging Mamba's internal structure.

5 Conclusion

We explore sparsification strategies within Mamba, revealing that selectively sparsifying the z-branch achieves superior accuracy-efficiency trade-offs compared to full sparsification or block-level approaches. The proposed Mamba-MoZ framework enhances performance while maintaining linear-time complexity, outperforming hybrid MoE-Mamba architectures. Our findings establish internal sparsification as a promising direction for scaling SSM-based models, paving the way for more efficient large-scale language models.

Limitations

Our work is limited by evaluating models on a 650M activated parameter model and three benchmarks due to computational constraints, leaving the scalability to larger models and diverse NLP tasks uncertain. The sparsification strategy for linear layers is relatively simple, and more methods can be explored in the future. In addition, although the sparsification strategy does not increase the amount of computation, it will bring about an increase in memory. Therefore, in large-scale model training, it is necessary to design a reasonable expert parallel strategy.

Acknowledge

Ruobing Xie is supported by the Young Elite Scientists Sponsorship Program by CAST (2023QNRC001).

References

Aaron Blakeman, Aarti Basant, Abhinav Khattar, Adithya Renduchintala, Akhiad Bercovich, Aleksander Ficek, Alexis Bjorlin, Ali Taghibakhshi, Amala Sanjay Deshmukh, Ameya Sunil Mahabaleshwarkar, et al. 2025. Nemotron-h: A family of accurate and efficient hybrid mamba-transformer models. *arXiv preprint arXiv:2504.03624*.

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*.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y Wu, et al. 2024. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*.

Tri Dao and Albert Gu. 2024. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. In *International Conference on Machine Learning*, pages 10041–10071. PMLR.

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.

Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*.

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*.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2020. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*.

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*.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *International Conference on Learning Representations*.

Jan Ludziejewski, Jakub Krajewski, Kamil Adamczewski, Maciej Pióro, Michał Krutul, Szymon Antoniak, Kamil Ciebiera, Krystian Król, Tomasz Odrzygóźdź, Piotr Sankowski, et al. 2024. Scaling laws for fine-grained mixture of experts. In *Proceedings of the 41st International Conference on Machine Learning*, pages 33270–33288.

Maciej Pióro, Kamil Ciebiera, Krystian Król, Jan Ludziejewski, Michał Krutul, Jakub Krajewski, Szymon Antoniak, Piotr Miłoś, Marek Cygan, and Sebastian Jaszczur. 2024. Moe-mamba: Efficient selective state space models with mixture of experts. *arXiv preprint arXiv:2401.04081*.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4463–4473.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*.

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.

An Wang, Xingwu Sun, Ruobing Xie, Shuaipeng Li, Jiaqi Zhu, Zhen Yang, Pinxue Zhao, JN Han, Zhanhui Kang, Di Wang, et al. 2024. Hmoe: Heterogeneous mixture of experts for language modeling. *arXiv preprint arXiv:2408.10681*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Hao Zheng, Zhanlei Yang, Wenju Liu, Jizhong Liang, and Yanpeng Li. 2015. Improving deep neural networks using softplus units. In *2015 International joint conference on neural networks (IJCNN)*, pages 1–4. IEEE.

A Introduction of Mamba Architecture

The original Mamba (Gu and Dao, 2023) framework operates through discrete-time parameterization of continuous SSMs. Given an input sequence $\{\mathbf{x}_k\}$, its core computations can be summarized as:

$$\begin{aligned}
\Delta_k &= \sigma\left(\mathbf{x}_k \mathcal{W}_\Delta + b_\Delta\right) \\
\mathbf{A}_k &= e^{-\Delta_k e^{\log \mathcal{W}_\mathbf{A}}} \\
\mathbf{B}_k &= \delta\left(\mathbf{x}_k \mathcal{W}_\mathbf{B}\right) \\
\mathbf{C}_k &= \delta\left(\mathbf{x}_k \mathcal{W}_\mathbf{C}\right) \\
\mathbf{h}_0 &= \mathbf{B}_0 \Delta_0 \mathbf{x}_0 \\
\mathbf{h}_k &= \overline{\mathbf{A}_{k-1}} \mathbf{h}_{k-1} + \mathbf{B}_k \Delta_k \mathbf{x}_k \\
\mathbf{y}_k &= \mathbf{C}_k \mathbf{h}_k \\
\mathbf{o}_k &= \mathbf{y}_k \times \delta(\mathbf{x}_k \mathcal{W}_\mathbf{z})
\end{aligned} \tag{2}$$

Here, σ denotes the softplus (Zheng et al., 2015) activation function, and δ represents the SiLU (Sigmoid Linear Unit) function. The architecture features input-dependent parameterization through Δ_k, which modulates both the discretization rate and the effective state transition.