# MoE-Mamba: Efficient Selective State Space Models with Mixture of Experts

**Maciej Pióro**
IPPT PAN
IDEAS NCBR

**Kamil Ciebiera**
University of Warsaw
IDEAS NCBR

**Krystian Król**
University of Warsaw
IDEAS NCBR

**Jan Ludziejewski**
University of Warsaw
IDEAS NCBR

**Michał Krutul**
University of Warsaw
IDEAS NCBR

**Jakub Krajewski**
University of Warsaw
IDEAS NCBR

**Szymon Antoniak**
IDEAS NCBR

**Piotr Miłoś**
University of Warsaw
IDEAS NCBR
IMPAN

**Marek Cygan**
University of Warsaw
Nomagic

**Sebastian Jaszczur**
University of Warsaw
IDEAS NCBR

## Abstract

State Space Models (SSMs) have become serious contenders in the field of sequential modeling, challenging the dominance of Transformers. At the same time, Mixture of Experts (MoE) has significantly improved Transformer-based Large Language Models, including recent state-of-the-art open models. We propose that to unlock the potential of SSMs for scaling, they should be combined with MoE. We showcase this on Mamba, a recent SSM-based model that achieves remarkable performance. Our model, MoE-Mamba, outperforms Mamba and matches the performance of Transformer-MoE. In particular, MoE-Mamba reaches the same performance as Mamba in $2.35\times$ *fewer training steps* while preserving the inference performance gains of Mamba against Transformer.

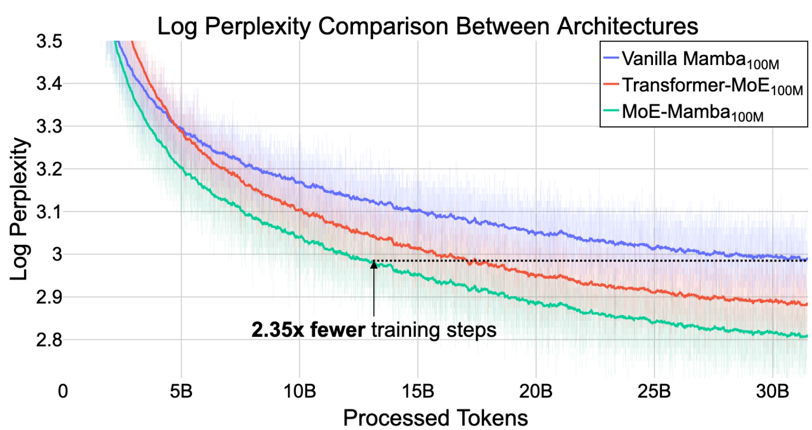


Figure 1: Log perplexity throughout the training. From top to bottom: $\text{Mamba}_{\text{100M}}$; $\text{Transformer-MoE}_{\text{100M}}$; $\text{MoE-Mamba}_{\text{100M}}$.

Correspondence to Sebastian Jaszczur <s.jaszczur@uw.edu.pl>.

## 1 INTRODUCTION

Large Language Models (LLMs) have emerged as a cornerstone in the ongoing AI revolution (Brown et al., 2020; Chowdhery et al., 2023; Lewkowycz et al., 2022; OpenAI, 2023; Team, 2023). Their remarkable effectiveness is primarily attributed to the Transformer architecture (Vaswani et al., 2017) and training on an internet-wide scale, e.g., (TogetherComputer, 2023). Yet, questions remain: Should Transformers be the only architecture used for LLMs? Can we scale language models even further, and if so, how can this be achieved?

Regarding the first question, State Space Models (SSMs), e.g., Gu et al. (2022b; 2021); Gupta et al. (2022); Li et al. (2022); Ma et al. (2022); Smith et al. (2023), have been increasingly gaining attention. Notably, a recent addition to this category, Mamba (Gu & Dao, 2023), has shown impressive results, positioning it as a promising contender to the attention-based Transformer architecture. Scaling is believed to be a critical factor in developing powerful AI systems (Sutton, 2019). The Mixture of Experts (MoE) approach (Jacobs et al., 1991), a set of techniques that enables an increase in model parameters with minimal impact on computational demands, plays a significant role. Due to their sparse activation, MoEs can be efficiently scaled up to trillions of parameters, as demonstrated by Fedus et al. (2022).

In this paper, *we advocate that to unlock the potential of SSMs for scaling up, they should be combined with MoE*. To this end, we introduce **MoE-Mamba**, combining Mamba (Gu & Dao, 2023) with a Switch layer (Fedus et al., 2022) and enabling efficiency gains of both SSMs and MoE. We confirm that the effect is robust to various design choices. In summary, our contributions are as follows:

- We introduce MoE-Mamba, a model that combines Mamba with a Mixture of Experts layer. MoE-Mamba enables efficiency gains of both SSMs and MoE while reaching the same performance as Mamba in $2.35\times$ fewer training steps, see Figure 1.
- Via comprehensive studies, we confirm that the improvement achieved by MoE-Mamba is robust to varying model sizes, design choices, and the number of experts.
- We explore and compare multiple alternative methods of integrating Mixture of Experts within the Mamba block.

## 2 RELATED WORK

**State Space Models** State Space Models (SSMs) (Gu et al., 2021; 2022a; Gupta et al., 2022; Li et al., 2022; Ma et al., 2022; Orvieto et al., 2023) form a family of architectures used for sequence modeling. Stemming from signal processing, these models can be seen as a combination of RNNs and CNNs (Gu & Dao, 2023). Recent breakthroughs (Gu et al., 2022b; Fu et al., 2023; Smith et al., 2023; Gu & Dao, 2023), have allowed deep SSMs to be increasingly competitive against Transformers (Vaswani et al., 2017). In particular, Mamba (Gu & Dao, 2023), studied in this paper, has shown impressive results through its selective mechanism and hardware-aware design, which allows scaling to billions of parameters while retaining computational efficiency and strong performance.

**Mixture of Experts** Mixture of Experts (MoE) is a class of techniques that allow drastically increasing the number of parameters of a model without much impact on the FLOPs required for the model's execution. Introduced in Jacobs et al. (1991); Jordan & Jacobs (1993), MoE was applied in the context of NLP by Shazeer et al. (2017). In MoE for each token processed, only a subset of the model's parameters is used. Due to their computational demands, feed-forward layers in Transformers have become the standard target of various MoE techniques (Lepikhin et al., 2020; Fedus et al., 2022; Du et al., 2022; Zoph et al., 2022). More recently, MoE models have found their way onto the open scene (Xue et al., 2023). In particular, the Mixtral $8\times7$B model (Jiang et al., 2024) fares comparably to Llama 2 70B (Touvron et al., 2023) while requiring only around $1/6$ of its inference computational budget.

## 3 MOE-MAMBA ARCHITECTURE

The vanilla Mamba architecture consists of multiple Mamba blocks stacked one after another, with each layer's output being added to the residual stream; see Figure 2. In MoE-Mamba, we replace

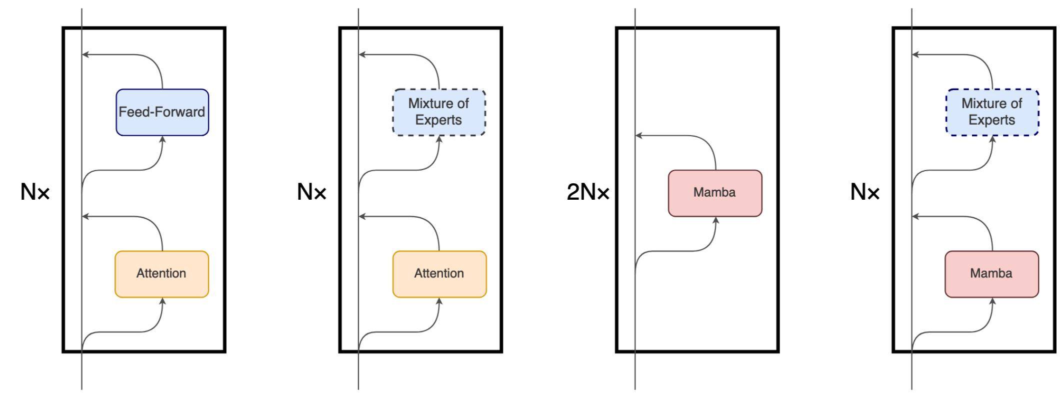


Figure 2: Diagrams of the architectures. From the left: vanilla Transformer, Transformer-MoE, Mamba, MoE-Mamba.

Table 1: Comparison between different architectures. The $\square_{25M}$ models were trained on ca. 10B tokens and the $\square_{100M}$ models were trained on ca. 30B tokens. Note that the parameter counts exclude embedding and output (unembedding) layers (for further discussion of reporting either non-embedding or all parameters, see Appendix E). The numbers of total and active parameters are not matched exactly between similarly sized models due to, among other reasons, the MoE models including routers and Mamba layer not containing precisely $6d_{\text{model}}^2$ parameters - a design choice we did not want to modify. We consider those differences to be too small to be significant for our results.

| **Model** | **# Parameters** | **# Active Parameters per Token** | **Final Log Perplexity** | **Speedup Over Vanilla Mamba (Training Steps)** |
|---|---|---|---|---|
| Mamba$_{25M}$ | 27M | 27M | 3.34 | 1 |
| MoE-Mamba$_{25M}$ (ours) | 542M | 26M | **3.19** | **1.76** |
| Transformer-MoE$_{25M}$ | 545M | 25M | 3.23 | 1.56 |
| Transformer$_{25M}$ | 25M | 25M | 3.43 | >1 |
| Mamba$_{100M}$ | 121M | 121M | 2.99 | 1 |
| MoE-Mamba$_{100M}$ (ours) | 2439M | 117M | **2.81** | **2.35** |
| Transformer-MoE$_{100M}$ | 2454M | 114M | 2.88 | 1.79 |

every other Mamba layer with a MoE layer (see Figure 2). We use the well-established (Zhao et al., 2023a) and easy-to-implement Switch Transformer MoE layer (Fedus et al., 2022) (for details, see Appendix B). This way, in MoE-Mamba, we separate unconditional processing of every token by the Mamba layer and conditional processing by an MoE layer. The idea of interleaving conditional and unconditional processing is used in some MoE-based models, typically by alternating vanilla and MoE feed-forward layers (Lepikhin et al., 2020; Fedus et al., 2022).

# 4 Experiments

## 4.1 Training Setup

We compare MoE-Mamba to three baselines: Mamba, Transformer, and Transformer-MoE. To be able to compare MoE-Mamba to Transformer-based and Mamba baselines, we scale down the size of each expert in our model as compared to traditional MoE approaches (we set $d_{\text{expert}} = 3d_{\text{model}}$ instead of $4d_{\text{model}}$), keeping the number of blocks and the number of active parameters per token roughly the same in all models of similar size. Active parameters denote those used to calculate the output for a given token (e.g., typically, only one expert in each MoE layer is active). For a discussion of the relation of active parameters and FLOPs, see Appendix C.

We train decoder-only models on the task of next token prediction using cross entropy as the loss function. For further details, refer to Appendix A. Due to computational constraints, we perform most of our experiments on smaller, $\square_{25\mathrm{M}}$ models and validate our findings on $\square_{100\mathrm{M}}$ models.

### 4.2 Main Results

Table 1 presents the comparison between training results of MoE-Mamba and baselines; see also Figure 1 for log perplexity curves. MoE-Mamba shows a remarkable improvement over the vanilla Mamba model. Notably, MoE-Mamba$_{100\mathrm{M}}$ was able to achieve the same performance as vanilla Mamba$_{100\mathrm{M}}$ with $2.35\times$ speedup in terms of processed tokens, similar to Transformer-MoE$_{100\mathrm{M}}$, strengthening the findings of Gu & Dao (2023) that Mamba is a competitive alternative to the Transformer. For $\square_{25\mathrm{M}}$ model size, the performance gains are even higher, however in Mamba$_{100\mathrm{M}}$, the gains might have been greater when trained on a larger number of tokens. For a detailed discussion of the speedup, see Appendix D.

### 4.3 Ablations

**Number of Experts** We investigate the impact of the number of experts used in Switch layers on MoE-Mamba and find that our approach scales favorably with the number of experts. MoE-Mamba outperforms vanilla Mamba, when the number of experts is $N_{\text{experts}} \geq 4$. This is consistent with Gu & Dao (2023) reporting that Mamba interleaved with feed-forward layers (which corresponds to a single-expert MoE layer) is worse than vanilla Mamba. We obtain the best result with the highest investigated expert count (32) and expect further gains with even more experts. For detailed results, see Appendix G.

**Optimal Ratio of Active Parameters in Mamba and MoE** In this section, we investigate the optimal ratio of active parameters in the Mamba layer to active parameters in the MoE layer while keeping the total number of parameters fixed. The results are presented in Figure 3 (left graph). We observe that increasing the number of active Mamba parameters improves the performance. However, the gains become marginal after reaching the $3:3$ ratio, and higher ratios are impractical due to inefficient hardware utilization and high routing costs caused by a large number of experts. We default to this choice in all other experiments. More details on selecting the optimal ratio can be found in Appendix F.

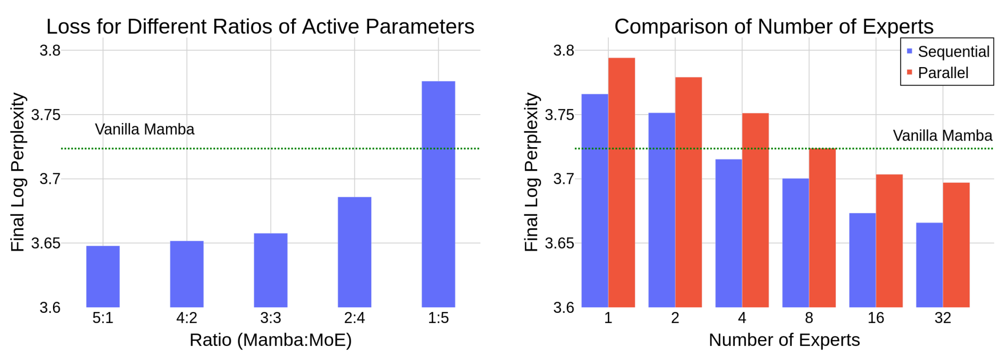


Figure 3: Left: Final loss at different ratios of active Mamba-to-MoE parameters. Note that MoE contains the majority of the total parameters in each model. For further discussion of the ratios explored, see Appendix F. Right: Final loss varying number of experts in sequential and parallel MoE-Mamba.

**Parallel MoE-Mamba** Inspired by Wang (2021) and Chowdhery et al. (2023), we experiment with an alternative block design in which the MoE feed-forward layer and the Mamba layer are placed in parallel instead of sequentially (see Figure 6 in Appendix). We compare this design to MoE-Mamba for various numbers of experts; see Figure 3 (right). MoE-Mamba outperforms this variant in all tested settings. The parallel MoE-Mamba matches vanilla Mamba when $N_{\text{experts}} \geq 8$ while requiring between 2 and 4 times as many experts and total parameters to match the performance of the sequential variant.

### 4.4 INNER MOE

Pursuing a uniform layer design, we experimented with replacing each of the three linear projections within the Mamba block with an MoE layer. Inspired by Fedus et al. (2022), we also performed experiments in which only half of the Mamba blocks were modified to include MoE. For more details on the experiments, see Appendix H. Three of the designs (Table 7 in Appendix) achieved results marginally better than vanilla Mamba, with none outperforming MoE-Mamba. These results suggest the most promising research directions for future work.

## 5 CONCLUSIONS

In this work, we present the first integration of Mixture of Experts with Mamba architecture, MoE-Mamba. This novel method inherits the inference benefits of Mamba and MoE while requiring $2.35\times$ fewer training steps to reach the same performance as Mamba. We also investigate the impact of the number of experts on the performance and explore numerous alternative design choices.

Our work opens a new research direction of combining Mixture of Experts with State Space Models. We believe that this path will enable more efficient scaling to even larger language models.

## ACKNOWLEDGMENTS

This work was funded by IDEAS NCBR, which also provided significant computational resources a supportive research environment and direction. The research was supported by PL-Grid infrastructure (grant PLG/2023/016148). We acknowledge snakes and experts as essential to our work. We also benefited from the Entropy cluster (hosted at the Faculty of Mathematics, Informatics and Mechanics of the University of Warsaw) funded by NVIDIA, Intel, the Polish National Science Center grant 2022/45/N/ST6/02222, and ERC Starting Grant TOTAL. Marek Cygan was partially supported by an NCBiR grant POIR.01.01.01-00-0392/17-00.

## REFERENCES

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020. URL `https://arxiv.org/abs/2005.14165`.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pp. 5547–5569. PMLR, 2022.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, 2022.

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

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

Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. Combining recurrent, convolutional, and continuous-time models with linear state-space layers, 2021.

Albert Gu, Karan Goel, Ankit Gupta, and Christopher Ré. On the parameterization and initialization of diagonal state space models. *Advances in Neural Information Processing Systems*, 35:35971–35983, 2022a.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces, 2022b.

Ankit Gupta, Albert Gu, and Jonathan Berant. Diagonal state spaces are as effective as structured state spaces. *Advances in Neural Information Processing Systems*, 35:22982–22994, 2022.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991. doi: 10.1162/neco.1991.3.1.79.

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, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts, 2024.

M.I. Jordan and R.A. Jacobs. Hierarchical mixtures of experts and the em algorithm. In *Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan)*, volume 2, pp. 1339–1344 vol.2, 1993. doi: 10.1109/IJCNN.1993.716791.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations, 2020.

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

Aitor Lewkowycz, Anders Johan Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Venkatesh Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=IFXTZERXdM7`.

Yuhong Li, Tianle Cai, Yi Zhang, Deming Chen, and Debadeepta Dey. What makes convolutional models great on long sequence modeling? *arXiv preprint arXiv:2210.09298*, 2022.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

Xuezhe Ma, Chunting Zhou, Xiang Kong, Junxian He, Liangke Gui, Graham Neubig, Jonathan May, and Luke Zettlemoyer. Mega: moving average equipped gated attention. *arXiv preprint arXiv:2209.10655*, 2022.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

OpenAI. Gpt-4 technical report, 2023.

Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. *arXiv preprint arXiv:2303.06349*, 2023.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, Kranthi Kiran GV, Xuzheng He, Haowen Hou, Jiaju Lin, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartlomiej Koptyra, Hayden Lau, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Guangyu Song, Xiangru Tang, Bolun Wang, Johan S. Wind, Stanislaw Wozniak, Ruichong Zhang, Zhenyuan Zhang, Qihang Zhao, Peng Zhou, Qinghua Zhou, Jian Zhu, and Rui-Jie Zhu. Rwkv: Reinventing rnns for the transformer era, 2023.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017.

Jimmy T. H. Smith, Andrew Warrington, and Scott W. Linderman. Simplified state space layers for sequence modeling, 2023.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023.

Richard Sutton. The bitter lesson. *Incomplete Ideas (blog)*, 13(1), 2019.

Gemini Team. Gemini: A family of highly capable multimodal models, 2023.

TogetherComputer. Redpajama: An open source recipe to reproduce llama training dataset, 2023. URL `https://github.com/togethercomputer/RedPajama-Data`.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Well-read students learn better: On the importance of pre-training compact models, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

Ben Wang. Mesh-Transformer-JAX: Model-Parallel Implementation of Transformer Language Model with JAX. https://github.com/kingoflolz/mesh-transformer-jax, May 2021.

Fuzhao Xue, Zian Zheng, Yao Fu, Jinjie Ni, Zangwei Zheng, Wangchunshu Zhou, and Yang You. Openmoe: Open mixture-of-experts language models. https://github.com/XueFuzhao/OpenMoE, 2023.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models, 2023a.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. Pytorch fsdp: Experiences on scaling fully sharded data parallel, 2023b.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew Dai, Zhifeng Chen, Quoc Le, and James Laudon. Mixture-of-experts with expert choice routing, 2022.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. St-moe: Designing stable and transferable sparse expert models, 2022.

## A HYPERPARAMETERS AND TRAINING SETUP

We train the models on C4 dataset Raffel et al. (2020) on the next token prediction task using cross entropy as the loss function. We process only a small fraction of the training set, allowing us to use EMA-smoothed ($\alpha = 0.001$) training log perplexity as the comparison metric for both final loss and speedup measurements. All models use the GPT2 tokenizer Radford et al. (2019). We tune the learning rate separately for all $\square_{\text{25M}}$ models and divide it by 2 when training their $\square_{\text{100M}}$ counterparts. The main experiments, described in section 4.2, use around 30B tokens (10B for $\square_{\text{25M}}$ models), while the experiments described in further sections use 1B tokens.

Basic model hyperameters ($d_{\text{model}}$, $d_{\text{ff}}$, the number of attention heads, the number of layers) used in this work were inspired by BERT (Devlin et al., 2019; Turc et al., 2019), with the $\square_{\text{25M}}$ models being equivalent to $\text{BERT}_{\text{MEDIUM}}$ and $\square_{\text{100M}}$ models copying $\text{BERT}_{\text{BASE}}$ configuration while increasing the number of blocks from 12 to 16. The learning rate schedule, as well as weight decay and gradient clipping values were set per community's standard practices. We used the AdamW optimizer (Loshchilov & Hutter, 2019). We tune the maximum learning rate value for each of the $\square_{\text{25M}}$ models separately and divide it by 2 when training $\square_{\text{100M}}$ counterparts. We train the models using PyTorch (Paszke et al., 2019) and utilize FSDP (Zhao et al., 2023b) for facilitating multi-GPU setup.

Table 2: Hyperparameters ($\square_{\text{25M}}$ Models). In Transformer models we use Rotary Position Embedding (Su et al., 2023).

| **Hyperparameter** | | **Transformer$_{\text{25M}}$** | **Mamba$_{\text{25M}}$** | **Transformer-MoE$_{\text{25M}}$** | **MoE-Mamba$_{\text{25M}}$** |
|---|---|---|---|---|---|
| Model | Total Blocks | 8 | 16 | 8 | 8 |
| | $d_{\text{model}}$ | 512 | 512 | 512 | 512 |
| | # Parameters | 25M | 27M | 545M | 542M |
| | # Active Parameters per Token | 25M | 27M | 25M | 26M |
| Feed-Forward | $d_{\text{ff}}$ | 2048 | - | - | - |
| Mixture of Experts | $d_{\text{expert}}$ | - | - | 2048 | 1536 |
| | $N_{\text{experts}}$ | - | - | 32 | 42 |
| Position Embedding | | RoPE | - | RoPE | - |
| Attention | $N_{\text{heads}}$ | 8 | - | 8 | - |
| Training | Training Steps | 150K | 150K | 150K | 150K |
| | Context Length | 1024 | 1024 | 1024 | 1024 |
| | Batch Size | 64 | 64 | 64 | 64 |
| | Max Learning Rate | 5e-4 | 1e-3 | 5e-4 | 5e-4 |
| | LR Warmup | 1% | 1% | 1% | 1% |
| | LR Schedule | Cosine | Cosine | Cosine | Cosine |
| | Final LR Ratio | 0.1 | 0.1 | 0.1 | 0.1 |
| | Weight Decay | 0.1 | 0.1 | 0.1 | 0.1 |
| | Gradient Clipping | 0.5 | 0.5 | 0.5 | 0.5 |

## B SWITCH MOE LAYER

In each Switch MoE layer, we assume $N_{\text{experts}}$ experts $\{E_i\}_{i=1}^{N_{\text{experts}}}$, each being a trainable feed-forward network with the same number of parameters. For each token embedding $x$, we calculate scores $h(x) = Wx \in \mathbb{R}^{N_{\text{experts}}}$, where $W$ is a trainable linear projection. These are normalized using softmax:

$$p_i(x) = \frac{\exp(h(x)_i)}{\sum_{i=1}^{N_{\text{experts}}} \exp(h(x)_i)}.$$

Prior to Switch, top-$k$ routing selecting $k > 1$ most suitable experts for each token was deemed necessary. However, Switch successfully simplifies previous MoE approaches by setting $k = 1$. Namely, the output of the MoE layer for $x$ is given by:

$$y = p_I E_I(x),$$

where $I = \arg\max_i p_i(x)$.

Table 3: Hyperparameters ($\square_{\text{100M}}$ Models). In Transformer-MoE$_{\text{100M}}$ we use Rotary Position Embedding Su et al. (2023).

| Hyperparameter | | Mamba$_{\text{100M}}$ | Transformer-MoE$_{\text{100M}}$ | MoE-Mamba$_{\text{100M}}$ |
|---|---|---|---|---|
| Model | Total Blocks | 32 | 16 | 16 |
| | $d_{\text{model}}$ | 768 | 768 | 768 |
| | # Parameters | 121M | 2454M | 2439M |
| | # Active Parameters per Token | 121M | 114M | 117M |
| Mixture of Experts | $d_{\text{expert}}$ | - | 3072 | 2304 |
| | $N_{\text{experts}}$ | - | 32 | 42 |
| Position Embedding | | - | RoPE | - |
| Attention | $N_{\text{heads}}$ | - | 12 | - |
| Training | Training Steps | 30K | 30K | 30K |
| | Context Length | 1024 | 1024 | 1024 |
| | Batch Size | 1024 | 1024 | 1024 |
| | Max Learning Rate | 1e-3 | 2.5e-4 | 5e-4 |
| | LR Warmup | 1% | 1% | 1% |
| | LR Schedule | Cosine | Cosine | Cosine |
| | Final LR Ratio | 0.1 | 0.1 | 0.1 |
| | Weight Decay | 0.1 | 0.1 | 0.1 |
| | Gradient Clipping | 0.5 | 0.5 | 0.5 |

Table 4: Comparison of sequential and parallel MoE-Mamba - final log perplexity (1B tokens).

| # of Experts | MoE-Mamba | |
|---|---|---|
| | Sequential | Parallel |
| 1 | 3.76 | 3.79 |
| 2 | 3.74 | 3.77 |
| 4 | 3.71 | 3.74 |
| 8 | 3.69 | 3.72 |
| 16 | 3.67 | 3.70 |
| 32 | **3.66** | 3.69 |

During batched execution, e.g., in training, each batch contains $N$ tokens. Following the standard procedure, in a case where the assignment of tokens to the experts is not perfect, i.e., some expert $E_f$ is selected by more than $N/N_{\text{experts}}$ tokens in the current batch, the excess tokens are dropped and not updated (capacity factor $= 1$). To further encourage an even distribution of tokens to experts, load balancing loss as described by Fedus et al. (2022) with weight $\alpha = 0.01$ is added to the training objective.

## C Active Parameters vs FLOPs

In this work, we report the number of active parameters (excluding embedding and unembedding layers) and not the number of floating-point operations (FLOPs), following Zhou et al. (2022). Both numbers will be roughly proportional (Kaplan et al., 2020), but the number of FLOPs is both harder to calculate and less relevant for hardware-aware architecture like Mamba with its optimizations, especially during inference.

## D RELATION BETWEEN SPEEDUP AND TRAINING TIME

In our experiments, we notice that as the training continues, the speedup of MoE-Mamba compared to vanilla Mamba generally increases (see Fig. 4). That is, the ratio

$$\text{speedup}(l) = \frac{\text{\# processed tokens vanilla Mamba took to reach loss } l}{\text{\# processed tokens MoE-Mamba took to reach loss } l}$$

increases as $l$ decreases. Speedup in $\square_{\text{25M}}$ models oscillates between 1.6 and 1.9, while the speedup in $\square_{\text{100M}}$ models rises steadily.

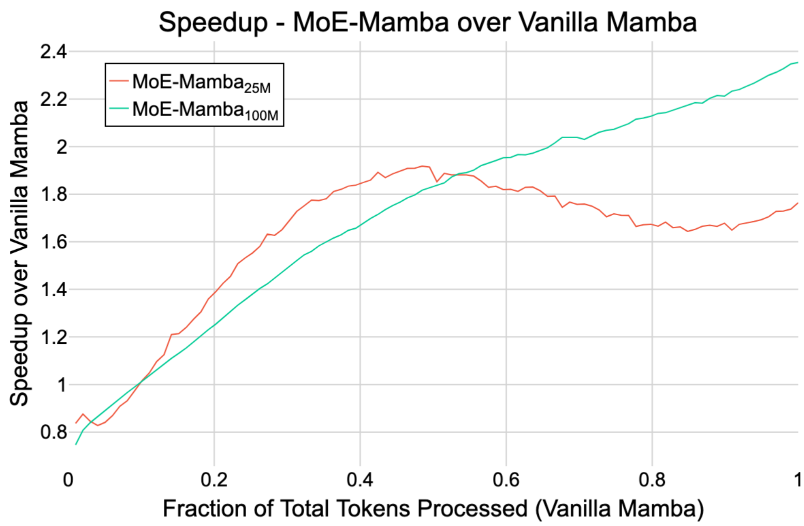


Figure 4: Speedup of different sizes of MoE-Mamba compared to their vanilla Mamba counterparts as training progresses.

## E COUNTING MODEL PARAMETERS

For all models and their variants, we report the number of trainable, non-embedding parameters, i.e., we exclude the parameters in the input (embedding) and output (unembedding) layers. This convention is proposed by Kaplan et al. (2020), who note that using just non-embedding parameters gives their scaling laws a clearer form. The relatively low importance of the number of the embedding parameters for the final performance has been noted by Lan et al. (2020).

## F EXPLORING THE OPTIMAL MAMBA TO MOE ACTIVE PARAMETERS RATIO

The assignment of FLOPs and parameters to different components is an important design choice in heterogeneous architectures. For example, in Transformer, the shape of the model has been studied extensively by Kaplan et al. (2020).

In our work, we investigate the optimal ratio of active parameters in the Mamba layer to the number of active parameters in the MoE layer. We vary the ratio while keeping $d_{\text{model}}$, the number of blocks and the total number of parameters fixed. Under these constraints, a given ratio determines the so-called expansion factor $E$ of the Mamba layer, the number of experts, and their size as detailed in Table 5 (see also Figure 6 for Mamba design). Figure 3 may suggest that increasing the ratio strengthens the performance and maybe assigning all the active parameters to Mamba would result in the best performance (ratio *"6:0"*). It should, however, be noted, that all the investigated models contain the same number of both total parameters and active parameters per token. A hypothetical model described above ("'6:0") could not achieve this property. If we loosen the requirements and place all the parameters in Mamba, lowering the umber of total parameters, the resulting model is the same as Mamba$_{\text{25M}}$ with the expansion factor $E = 4$ and 8 instead of 16 Mamba layers. This model achieves marginally worse final log perplexity than Mamba$_{\text{25M}}$ (3.73).

Table 5: Comparison of different ratios of parameters between Mamba and MoE. The $E = 2$ corresponds to MoE-Mamba$_{25M}$. The total number of parameters in all models is 542M and the number of active parameters per token is 26M.

| Ratio $N^{\text{act. params}}_{\text{Mamba}} : N^{\text{act. params}}_{\text{MoE}}$ | Expansion Factor $E$ (Mamba) | Expert Size | Number of Experts |
|---|---|---|---|
| $1:5$ | $\frac{2}{3}$ | 2560 | 19 |
| $2:4$ | $1\frac{2}{3}$ | 2048 | 24 |
| $3:3$ | $2$ | 1536 | 32 |
| $4:2$ | $2\frac{2}{3}$ | 1024 | 48 |
| $5:1$ | $3\frac{1}{3}$ | 512 | 96 |

## G Optimal Number of Experts

Figure 5 shows the training runs for different numbers of experts. The results show that our approach scales favorably with the number of experts. MoE-Mamba outperforms vanilla Mamba, when the number of experts is $N_{\text{experts}} \geq 4$. We obtain the best result with 32 experts and expect further gains with even more experts. Table 6 shows the final results.

Table 6: Log perplexity after 1B tokens for various numbers of experts. Note that the parameter counts exclude the embedding and output (unembedding) layers.

| **Number of Experts** | **# Parameters** | **# Active Parameters per Token** | **Log Perplexity After 1B Tokens** | **Speedup Over Vanilla Mamba (Training Steps)** |
|---|---|---|---|---|
| N/A - Vanilla Mamba | 27M | 27M | 3.72 | 1 |
| 1 | 26M | 26M | 3.75 | <1 |
| 4 experts | 64M | 26M | 3.72 | 1.03 |
| 8 experts | 114M | 26M | 3.70 | 1.10 |
| 16 experts | 215M | 26M | 3.67 | 1.21 |
| 32 experts | 416M | 26M | **3.67** | **1.23** |

## H Inner MoE

As described in Section 4.4, we experimented with replacing each of the three linear projections within the Mamba block with an MoE layer; see Figure 6. Enumerating all the possible placements results in $2^3 - 1 = 7$ possible designs (we discard one combination that would feature no MoE inside the block). We maintain a similar number of total parameters and FLOPs in all models by assuring the total number of expert feed-forward layers in a block sums up to $24$ regardless of the placement, i.e., the $24$ experts are split evenly between one, two or three MoE's inside the block. Inspired by Fedus et al. (2022), we also performed experiments in which only half of the Mamba blocks were modified to include MoE, but the number of experts was increased to $48$ to maintain the total number of parameters.

Three of the designs (Table 7) achieved results marginally better than vanilla Mamba, with none outperforming MoE-Mamba.

## I Accuracy and Perplexity

We have observed a curious case of metric inconsistency between two models that achieved similar performance but were based on different architectures, namely MoE-Mamba$_{25M}$ with 32 instead of 42 experts and Transformer-MoE$_{25M}$. We hypothesize that this discrepancy hints at a potential failure mode of Mamba and other SSMs. Due to the compression of the history into a finite hidden state,

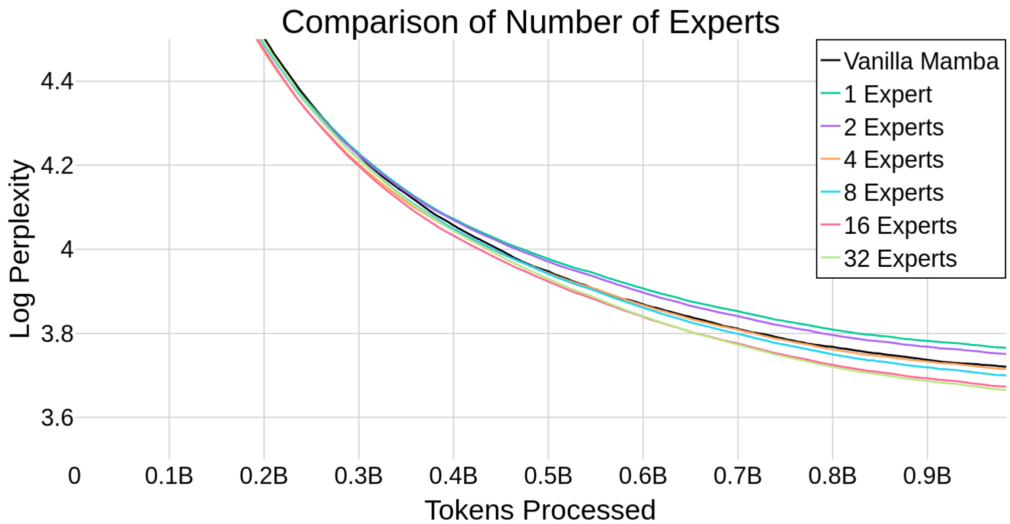


Figure 5: Training loss (log perplexity) for a differing number of experts for MoE-Mamba with ca. 26M active non-embedding parameters. The final log perplexity improves monotonically as the number of experts increases.

Table 7: Comparison of different variants of MoE in Mamba - final log perplexity (1B tokens).

| **Model Name / Modified Projection** | **MoE in Mamba** | |
|---|---|---|
| | All Layers | Every Other Layer |
| Vanilla Mamba | 3.72 | |
| MoE-Mamba (16 experts) | **3.67** | |
| Conv Projection | 3.79 | 3.71 |
| Gate Projection | 3.89 | 3.70 |
| Output Projection | 4.05 | 3.70 |
| Conv + Gate Projection | 3.95 | 3.72 |
| Conv + Output Projection | 4.17 | 3.76 |
| Gate + Output Projection | 4.16 | 3.88 |
| Conv + Gate + Output Projection | 4.39 | 3.88 |

their ability for verbatim token-copying is limited. The related ability to predict the token $[B]$ given a prefix ...$[A][B]$...$[A]$ (where $[A]$, $[B]$ can be any tokens) has been mechanistically studied by Elhage et al. (2021) and has been conjectured to be responsible for Transformer's remarkable in-context learning capabilities (Olsson et al., 2022).

Peng et al. (2023) mentions that their attention-free model, RWKV, may have limited performance on tasks that require recalling precise information over long contexts due to a fixed-sized hidden state, a property that Mamba and other SSMs share. However, since the perplexity of Mamba can match the perplexity of a similarly-sized Transformer, we can suspect that Mamba compensates for that failure mode in other ways and might show a relative advantage on other tasks when compared to Transformer. In particular, it might outperform Transformers in 0-shot tasks in contrast to tasks allowing few-shot demonstrations or requiring in-context learning.

## J Reproducibility

The codebase used to run the experiments is available at https://github.com/llm-random/llm-random.

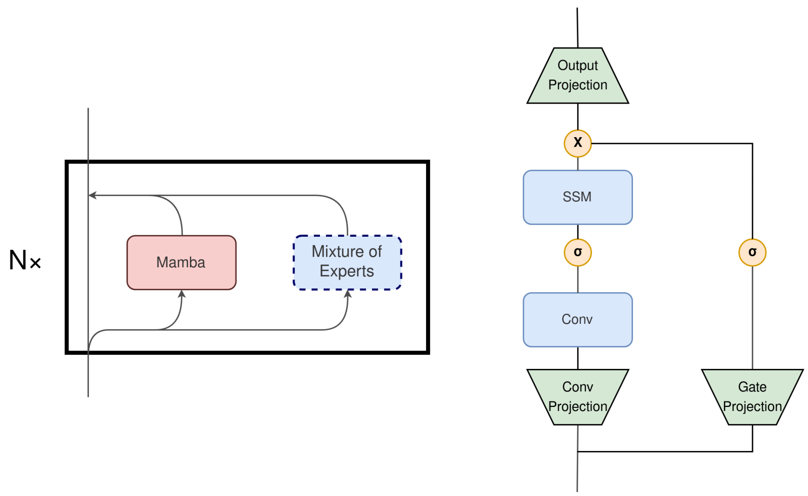


Figure 6: Diagram of Parallel MoE-Mamba architecture (left) and Mamba Block (right). The outputs of the Gate and Conv Projections are $E$ (expansion factor) times bigger than the input, i.e., Conv and SSM operate on vectors $\in \mathbb{R}^{E \cdot d_{\text{model}}}$. Vanilla Mamba assumes $E = 2$ (Gu & Dao, 2023). Expansion factor $E$ determines how much the input vector is scaled up by Gate and Conv Projection and then scaled down by Output Projection, and because of that, it is also proportional to the number of FLOPs and parameters in the Mamba layer.

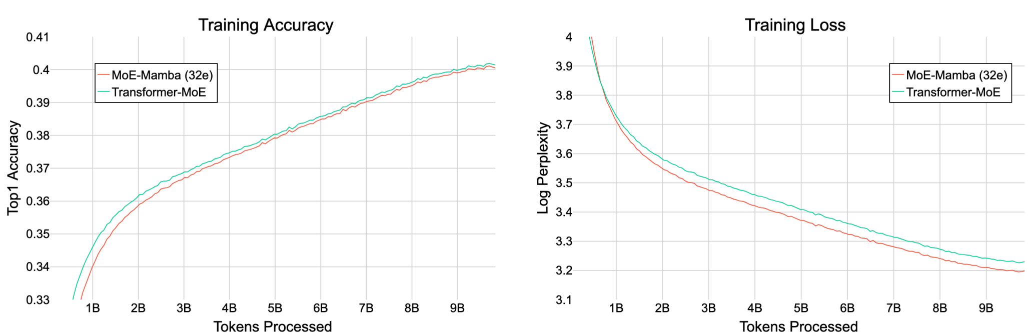


Figure 7: Discrepancy between accuracy and log perplexity: MoE-Mamba with 32 experts and Transformer-MoE.