Mixture of Sparse Attention: Content-Based Learnable Sparse Attention via Expert-Choice Routing

Piotr Piękos[1], **Róbert Csordás**[2], **Jürgen Schmidhuber**[1]
[1]KAUST, AI Initiative, Thuwal, Saudi Arabia
[2]Stanford University, Stanford, CA, USA
`piotr.piekos@kaust.edu.sa`

Abstract

Recent advances in large language models highlighted the excessive quadratic cost of self-attention. Despite the significant research efforts, subquadratic attention methods still suffer from inferior performance in practice. We hypothesize that dynamic, learned content-based sparsity can lead to more efficient attention mechanisms. We present Mixture of Sparse Attention (MoSA), a novel approach inspired by Mixture of Experts (MoE) with expert choice routing. MoSA dynamically selects tokens for each attention head, allowing arbitrary sparse attention patterns. By selecting k tokens from a sequence of length T, MoSA reduces the computational complexity of each attention head from $O(T^2)$ to $O(k^2 + T)$. This enables using more heads within the same computational budget, allowing higher specialization. We show that among the tested sparse attention variants, MoSA is the only one that can outperform the dense baseline, sometimes with up to 27% better perplexity for an identical compute budget. MoSA can also reduce the resource usage compared to dense self-attention. Despite using torch implementation without an optimized kernel, perplexity-matched MoSA models are simultaneously faster in wall-clock time, require less memory for training, and drastically reduce the size of the KV-cache compared to the dense transformer baselines.

1 Introduction

Modern transformer architectures [1] have proven to be highly effective for sequence modeling tasks and are the key to the success of large language models (LLMs; [2, 3, 4, 5]). The attention mechanism enables expressivity, but requires quadratic computational and memory complexity in terms of sequence length. This results in a high training and deployment cost of LLM. Furthermore, the KV-cache memory footprint during inference presents a significant bottleneck, limiting practical deployment and increasing operational costs.

To mitigate the attention costs, sparse attention methods have been investigated. Static sparse attention methods [6] reduce the quadratic complexity by selectively attending to a subset of tokens to be used in the attention. They use hand-defined coarse-grained patterns that are not data-dependent. Typical examples of these methods are the block-sparse and strided attention [6, 7, 8]. On other hand, content-based dynamic sparse attention [9, 10, 11] methods can, in principle, learn to attend to individual tokens, regardless of their location in the input, while ignoring less useful tokens. The Routing Transformer [11] clusters the tokens within each head using online K-means. However, it fails to show significant performance gains over static sparse-attention methods, possibly due to the slow convergence of online K-means [12].

Presented at the NeurIPS 2025 Workshop on Efficient Reasoning.

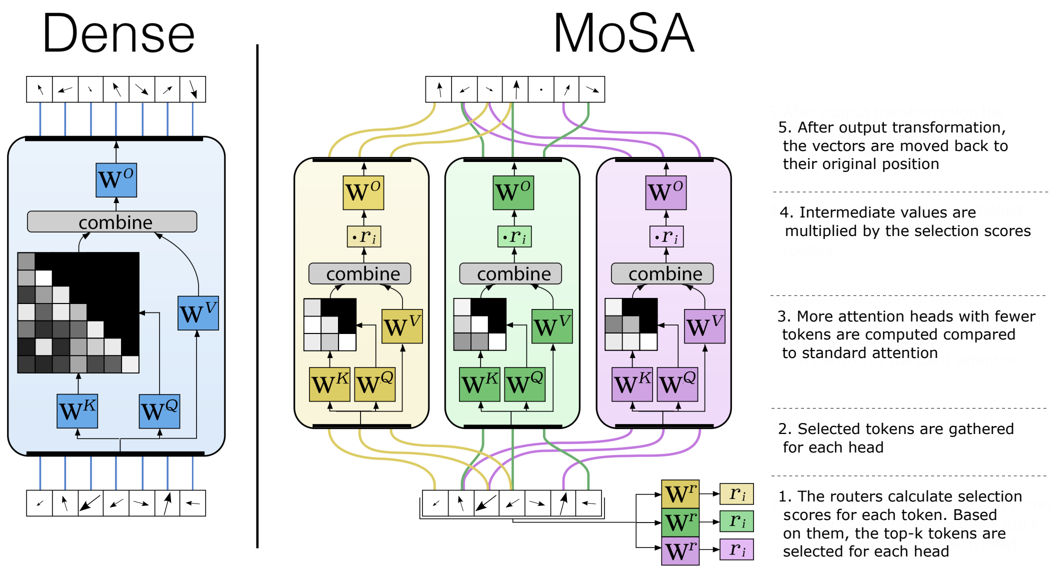
Dense
MoSA
combine
W^O
W^V
W^K
W^Q
$\cdot r_i$
W^r
r_i
5. After output transformation, the vectors are moved back to their original position
4. Intermediate values are multiplied by the selection scores
3. More attention heads with fewer tokens are computed compared to standard attention
2. Selected tokens are gathered for each head
1. The routers calculate selection scores for each token. Based on them, the top-k tokens are selected for each head

Figure 1: MoSA layer compared to the dense attention layer. MoSA replaces each dense head with multiple heads with a learnable sparsity pattern. Each head selects its own k tokens to process. MoSA calculates query, key, and value projections only for the selected token and computes the attention only between them. It drops the rest of the tokens, leading to more efficient compute utilization. This reduces the computational and memory complexity on a sequence of length T from $O(T^2)$ to $O(k^2 + T)$. The saved compute budget can be used to scale up the number of heads.

We propose a novel approach, inspired by Mixture-of-Experts [13, 14], to create a dynamic, content-based, and head-specific selection of tokens for sparse attention. This is achieved with Expert-Choice Routing [15], where each attention head is treated like an expert and selects its own specific tokens from the input. This creates a perfectly balanced selection, avoiding the need for complicated regularization techniques. We name our approach Mixture-of-Sparse Attention(*MoSA*).

By selecting k tokens from a sequence of length T, MoSA reduces the computational complexity of the attention head from $O(T^2)$ to $O(k^2 + T)$. Sparse attention techniques have historically been employed out of necessity to manage long sequences that exceed available computational capacities. In contrast, we also explore the use of the saved computation budget for creating additional attention heads. Thus, in this setup, MoSA employs a large number of highly sparse attention heads, encouraging their specialization. We show that this allows for better utilization of the available compute budget and leads to substantially better iso-flop language modeling performance compared to dense attention and investigated sparse attention baselines.

Following observations of sparse attention methods [6, 11], we incorporate MoSA as part of a hybrid model with a different type of attention. Our main results demonstrate that hybrid models with many MoSA and four dense heads significantly improve the model's quality by up to 27% in an IsoFLOP setting. MoSA consistently improves perplexity across all model scales starting with baselines from 28M to 516M parameters. Furthermore, we demonstrate that for long sequences, MoSA combined with local attention clearly outperforms other analyzed sparse attention methods with a fixed budget.

2 Method

Sparse attention methods model global dependencies by selecting specific tokens that can attend to other specific tokens based on a hand-engineered set of rules [8, 7] or by blockwise aggregation of tokens [16]. Both of these families of methods impose the mixing of information during token aggregation, either explicitly or implicitly. Instead, we propose to select tokens adaptively for each head based on the input. Thus, a flexible set of important tokens can be kept around, creating content-based sparsity without strong information mixing. To achieve that, we take inspiration from Expert-Choice routing in MoEs. MoSA is shown in Fig. 1.

In MoSA, in addition to the standard projections, each head has an additional router that selects which tokens are used for that head. Formally, the router is defined using the weight matrix $\mathbf{W}^r \in \mathbb{R}^h$. Let $\mathbf{X} \in \mathbb{R}^{T\times h}$ be the T-long sequence of input tokens. The router calculates the selection scores for each token $\mathbf{r} = \sigma(\mathbf{X}\mathbf{W}^r) \in \mathbb{R}^T$. For σ we use the non-competitive sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ following observations from σ-MoE [17]. Subsequently, we use expert choice for the selection of tokens for each head:

$$\mathbf{r}^{topk}, I = TopK(\mathbf{r}, k)$$

where $TopK$ returns the highest k values of $\boldsymbol{r}$ called $\mathbf{r}^{topk} \in \mathbb{R}^k$, along with their indices $I \in \{0, ..., T-1\}^k$. I is used to select the subset of inputs for the MoSA head:

$$\mathbf{X}^s = (\mathbf{X}_{I_1}, \mathbf{X}_{I_2}, ..., \mathbf{X}_{I_k}) \in \mathbb{R}^{k\times h}$$

where $\mathbf{X}_i$ represents $i'th$ row from matrix $\mathbf{X}$. After that, queries, keys, and values are calculated identically to the standard MHA: $\mathbf{X}^s$ as $\mathbf{Q} = \mathbf{X}^s\mathbf{W}^Q, \mathbf{K} = \mathbf{X}^s\mathbf{W}^K, \mathbf{V} = \mathbf{X}^s\mathbf{W}^V$. As our primary target is language modeling, we also calculate the mask that prohibits attending to future tokens. Unlike the standard MHA, this mask is not triangular and has to take into account the token indices selected by the head: $\mathbf{M}_{i,j} = 0 \iff I_i \geq I_j, -\infty$ otherwise.

The sparse attention can be computed using the standard attention defined in Eq. 2. $\mathbf{A} = \text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{M})$. This allows the combination of MoSA with optimized attention implementations such as Flash Attention [18]. The resulting vectors $\mathbf{A}_i$ are multiplied by the corresponding router values $\mathbf{r}_i$. Then, after the output transformation $\mathbf{W}^o$, they are moved back to their original positions in the full-length sequence $\mathbf{Y} \in \mathbb{R}^{T\times h}$.

$$\mathbf{X}^o = \text{diag}(\mathbf{r})\mathbf{A}\mathbf{W}^o \in \mathbb{R}^{k\times h}$$

$$\mathbf{Y}_j = \begin{cases} \mathbf{X}^o_i, & \text{if } j = I_i \text{ for some } i \in \{1, \dots, k\}, \\ 0, & \text{otherwise,} \end{cases} \quad \text{for } j = 1, \dots, T.$$

diag$(\cdot)$ creates a diagonal matrix from a vector, used for elementwise scaling of the columns of the matrix $\mathbf{A}$ by a vector $\mathbf{r}$. This ensures that the token's contribution is proportional to the router's output. This also enables the router to receive gradients, making it learnable by gradient descent. We call the combined transformation of x into y, parameterized by $\theta_i = (\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V, \mathbf{W}^O, \mathbf{W}^r)$ a single MoSA head: $\mathbf{Y} = \text{MoSA}_{head}(\mathbf{X}; \theta_i)$. A MoSA layer parameterized by $\theta = \{\theta_i\}_{i\in 1\dots H}$ is:

$$\text{MoSA}(\mathbf{X}; \theta) = \sum_{i=1}^{H} \text{MoSA}_{head}(\mathbf{X}; \theta_i) \tag{1}$$

The entire transformation in the multihead version can be efficiently implemented in PyTorch [19] using `einsum`, `scatter` and `gather` operations.

Sparse attention methods are usually combined with local attention [6, 11] when used on long sequences. Sparse attention then captures global dependencies, while local attention preserves local context. As our setup permits the use of dense attention, in our main experiments, we combine MoSA or corresponding sparse attention baseline with 4 dense heads. In Appendix E, we demonstrate the necessity of hybridization and motivate our selection of four dense heads for the models. In App. G, we combine MoSA with local attention for long sequences and demonstrate that MoSA demonstrates superior performance in this scenario as well.

3 Experiments

We empirically demonstrate MoSA's performance in different settings by comparing MoSA to dense and sparse baselines. In this section, we evaluate all the methods on language modeling under a fixed FLOP budget. In App. F we demonstrate the practical benefits of MoSA by measuring wall-clock time, memory usage, and KV cache size in a perplexity-matched setup. In App. G we investigate the performance of MoSA on long sequences. Furthermore, in Appendix H, we analyze the performance of different models in downstream zero-shot tasks.

Apart from a dense baseline, we compare MoSA with two sparse attention methods: Fixed sparse attention with strided selected tokens that participate in the attention and Routing Attention that represents a content-based sparse attention baseline. The baselines are described in detail and compared with MoSA in App. B and visualized in Fig. 3. The models'details are described in App. J.

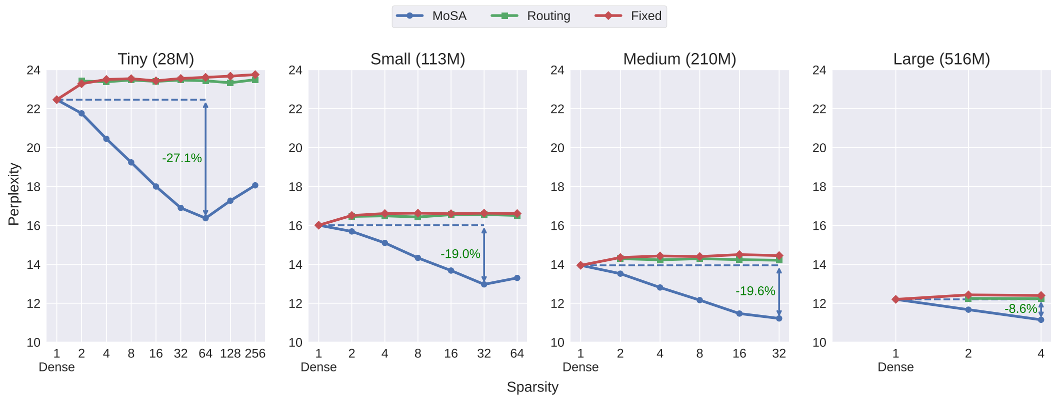

Figure 2: Perplexity (↓) of FLOP matched models under different sparsities. Each plot corresponds to a specified FLOP budget per step. The number in parenthesis is the number of parameters of the dense baseline. Sparsity 1 represents the dense baseline. As sparsity increases, MoSA's perplexity improves monotonically until reaching a saturation point around sparsity 32-64, beyond which performance deteriorates. This is likely because at very high sparsity levels, each attention head selects only a few tokens, which is insufficient to capture the complex relations. On the other hand, other sparse methods fail to reach the perplexity of the dense baseline in the IsoFLOP setting.

We evaluate sparse methods by gradually increasing sparsity rate $\rho = \frac{T}{k}$. This reduces the compute requirements for each sparse head. We use the saved budget to increase the number of sparse heads. Specifically, we choose the number of sparse heads to be the maximum such that the FLOPs of the sparse model do not exceed the FLOPs of the baseline model for a given size. All sparse models include four dense heads that we keep (see App. E), and are included in the FLOP calculations. Note that increasing the number of attention heads also increases the memory requirements of all methods. Consequently, for the larger FLOP-matched models, we restricted the explored sparsity values to ensure that the models fit in the memory budget dictated by our hardware.

Starting from sparsity 1, which corresponds to the dense model, we gradually increase the sparsity and measure the test-set perplexity of FLOP-matched models. Figure 2 illustrates the results for each model class and size. Across all model sizes tested, All MoSA hybrids reduce the perplexity of the baseline, sometimes by 27%. In constrast to MoSA, the sparse baselines for all $\rho > 1$ perform worse than the dense baseline. They exhibit relatively constant, but worse, perplexity across different sparsity values, with only minor fluctuations that reveal no discernible trend.

4 Conclusions

This paper introduces Mixture of Sparse Attention (MoSA), a novel attention architecture that selectively focuses on the most relevant tokens for the attention head, redirecting saved compute to create additional heads. MoSA reduces the computational complexity of attention from $O(T^2)$ to $O(k^2 + T)$, where T is the sequence length and k is the number of selected tokens per head.

Unlike other sparse attention methods that primarily show benefits for extremely long sequences, MoSA delivers substantial performance gains even in standard-length contexts. MoSA significantly outperforms both dense attention and sparse methods like fixed attention or the Routing Transformer, achieving up to 27% perplexity improvement over dense baselines across models of different scales. MoSA can also be used to reduce the resource requirements of the models, including a more than 50% reduction in the KV-cache size. Additionally, our results indicate that MoSA maintains its superiority in long-sequence scenarios, outperforming other sparse attention methods in these contexts as well.

The efficiency and corresponding performance gains demonstrated by MoSA have significant implications for the design of adaptive architectures. MoSA or subsequent adaptive models stemming from MoSA can be used for reducing the training costs and environmental impact of large language models, potentially enabling more economical scaling while lowering energy consumption and carbon emissions. Given its versatility and performance advantages, we anticipate that MoSA will drive innovations in both transformer architecture research and industrial applications.

Acknowledgements

The authors would like to thank Firas Laakom and Nanbo Li for their comments on the manuscript.

For computer time, this research used Ibex managed by the Supercomputing Core Laboratory at King Abdullah University of Science & Technology (KAUST) in Thuwal, Saudi Arabia.

The research reported in this publication was supported by funding from King Abdullah University of Science and Technology (KAUST) - Center of Excellence for Generative AI, under award number 5940.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, pages 5998–6008, Long Beach, CA, USA, December 2017.

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, pages 1877–1901, 2020.

[3] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and efficient foundation language models. *Preprint arXiv:2302.13971*, 2023.

[4] Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[5] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[6] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[7] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 17283–17297, 2020.

[8] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv:2004.05150*, 2020.

[9] Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. Synthesizer: Rethinking self-attention for transformer models. In *Proc. Int. Conf. on Machine Learning (ICML)*, pages 10183–10192, 2021.

[10] Apoorv Vyas, Angelos Katharopoulos, and François Fleuret. Fast transformers with clustered attention. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 21665–21674, 2020.

[11] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics (TACL)*, 9:53–68, 2021.

[12] Leon Bottou and Yoshua Bengio. Convergence properties of the k-means algorithms. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, volume 7, 1994.

[13] 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. In *Int. Conf. on Learning Representations (ICLR)*, Toulon, France, April 2017.

[14] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research (JMLR)*, 23(1):5232–5270, 2022.

[15] Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. Mixture-of-experts with expert choice routing. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 7103–7114, 2022.

[16] Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, YX Wei, Lean Wang, Zhiping Xiao, et al. Native sparse attention: Hardware-aligned and natively trainable sparse attention. *arXiv preprint arXiv:2502.11089*, 2025.

[17] Róbert Csordás, Kazuki Irie, and Jürgen Schmidhuber. Approximating two-layer feedforward networks for efficient transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, November 2023.

[18] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, New Orleans, Louisiana, USA, December 2022.

[19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary 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. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, pages 8024–8035, Vancouver, Canada, December 2019.

[20] Qingru Zhang, Dhananjay Ram, Cole Hawkins, Sheng Zha, and Tuo Zhao. Efficient long-range transformers: You need to attend more, but not necessarily at every layer. In *Proc. Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[21] Nikita Kitaev, Kaiser Łukasz, and Anselm Levskaya. Reformer: The efficient transformer. In *Int. Conf. on Learning Representations (ICLR)*, 2020.

[22] Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J. Colwell, and Adrian Weller. Rethinking attention with performers. In *Int. Conf. on Learning Representations (ICLR)*, Virtual only, May 2021.

[23] Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to recurrent nets. *Neural Computation*, 4(1):131–139, 1992.

[24] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proc. Int. Conf. on Machine Learning (ICML)*, volume 119, pages 5156–5165, Virtual Only, 2020.

[25] Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *Proc. Int. Conf. on Machine Learning (ICML)*, volume 139, pages 9355–9366, Virtual only, 2021.

[26] Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. Hippo: Recurrent memory with optimal polynomial projections. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 1474–1487, 2020.

[27] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *Int. Conf. on Learning Representations (ICLR)*, 2022.

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

[29] Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Zou, Atri Rudra, and Christopher Ré. Zoology: Measuring and improving recall in efficient language models. *Int. Conf. on Learning Representations (ICLR)*, 2024.

[30] Samy Jelassi, David Brandfonbrener, Sham M Kakade, and Eran Malach. Repeat after me: Transformers are better than state space models at copying. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2024.

[31] Guoxuan Chen, Han Shi, Jiawei Li, Yihang Gao, Xiaozhe Ren, Yimeng Chen, Xin Jiang, Zhenguo Li, Weiyang Liu, and Chao Huang. Sepllm: Accelerate large language models by compressing one segment into one separator. *arXiv preprint arXiv:2412.12094*, 2024.

[32] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *Int. Conf. on Learning Representations (ICLR)*, 2024.

[33] Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pages 52342–52364, 2023.

[34] Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pages 22947–22970, 2025.

[35] Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Baobao Chang, Junjie Hu, and Xiao Wen. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*, 2024.

[36] Aditya Desai, Shuo Yang, Alejandro Cuadron, Ana Klimovic, Matei Zaharia, Joseph E Gonzalez, and Ion Stoica. Hashattention: Semantic sparsity for faster inference. *arXiv preprint arXiv:2412.14468*, 2024.

[37] 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. In *Int. Conf. on Learning Representations (ICLR)*, 2021.

[38] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[39] 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. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[40] Yikang Shen, Zhen Guo, Tianle Cai, and Zengyi Qin. Jetmoe: Reaching llama2 performance with 0.1 m dollars. *arXiv preprint arXiv:2404.07413*, 2024.

[41] Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. BASE layers: Simplifying training of large, sparse models. In Marina Meila and Tong Zhang, editors, *Proc. Int. Conf. on Machine Learning (ICML)*, volume 139, pages 6265–6274, Virtual only, July 2021.

[42] Stephen Roller, Sainbayar Sukhbaatar, Jason Weston, et al. Hash layers for large sparse models. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 17555–17566, 2021.

[43] David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024.

[44] Xiaofeng Zhang, Yikang Shen, Zeyu Huang, Jie Zhou, Wenge Rong, and Zhang Xiong. Mixture of attention heads: Selecting attention heads per token. In *Proc. Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4150–4162, Abu Dhabi, United Arab Emirates, December 2022.

[45] Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

[46] Róbert Csordás, Piotr Piękos, Kazuki Irie, and Jürgen Schmidhuber. Switchhead: Accelerating transformers with mixture-of-experts attention. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, Vancouver, Canada, December 2024.

[47] Peng Jin, Bo Zhu, Li Yuan, and Shuicheng Yan. Moh: Multi-head attention as mixture-of-head attention. *arXiv preprint arXiv:2410.11842*, 2024.

[48] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer

architecture. In *Proc. Int. Conf. on Machine Learning (ICML)*, volume 119, pages 10524–10533, Virtual Only, July 2020.

[49] Chejian Xu, Wei Ping, Peng Xu, Zihan Liu, Boxin Wang, Mohammad Shoeybi, Bo Li, and Bryan Catanzaro. From 128k to 4m: Efficient training of ultra-long context large language models. *arXiv preprint arXiv:2504.06214*, 2025.

[50] Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proc. Association for Computational Linguistics (ACL)*, Berlin, Germany, August 2016.

[51] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *Proc. AAAI Conf. on Artificial Intelligence*, pages 8732–8740, New York, NY, USA, February 2020.

[52] Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. BLiMP: The benchmark of linguistic minimal pairs for English. *Transactions of the Association for Computational Linguistics (TACL)*, 8:377–392, 2020.

[53] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proc. Association for Computational Linguistics (ACL)*, pages 4791–4800, Florence, Italy, August 2019.

[54] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: reasoning about physical commonsense in natural language. In *Proc. AAAI Conf. on Artificial Intelligence*, pages 7432–7439, New York, NY, USA, February 2020. AAAI Press.

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

[56] 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. *arXiv preprint arXiv:2202.08906*, 2022.

[57] Sheng Shen, Le Hou, Yanqi Zhou, Nan Du, Shayne Longpre, Jason Wei, Hyung Won Chung, Barret Zoph, William Fedus, Xinyun Chen, et al. Mixture-of-experts meets instruction tuning: A winning combination for large language models. In *Int. Conf. on Learning Representations (ICLR)*, 2024.

[58] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proc. Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4895–4901, 2023.

[59] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proc. Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, pages 66–71, Brussels, Belgium, October 2018.

[60] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proc. Association for Computational Linguistics (ACL)*, pages 1715–1725, Berlin, Germany, August 2016.

[61] Mike Schuster and Kaisuke Nakajima. Japanese and korean voice search. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152, Kyoto, Japan, March 2012.

[62] 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. *Journal of Machine Learning Research (JMLR)*, 21:140:1–140:67, 2020.

[63] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *Int. Conf. on Learning Representations (ICLR)*, San Diego, CA, USA, May 2015.

[64] Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Preprint arXiv:2104.09864*, 2021.

A Background

Attention Mechanism. Attention assigns input-dependent weights to tokens in a sequence, allowing each token to gather context from the rest of the sequence. To do this, each token is projected to three vectors: its *query*, *key*, and *value*. For a given token, we compare its *query* vector with the *key* vectors of all tokens (including itself), producing a set of similarity scores. The scores are then normalized and used to calculate a weighted sum of the tokens' *value* vectors. The result is a new representation that dynamically integrates information throughout the sequence.

Let T be the sequence length, h the hidden dimension of the model, and h' the hidden dimension in each head. $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{T \times h'}$ represents the query, key and value matrices, respectively.

The attention is computed as:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{M}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top + \mathbf{M}}{\sqrt{h'}}\right)\mathbf{V} \tag{2}$$

Here, $\mathbf{M}$ denotes the attention mask that represents hard modeling constraints. $\mathbf{M}_{i,j} = 0$ if and only if $i'th$ token is allowed to attend to $j'th$ token, otherwise $\mathbf{M}_{i,j} = -\infty$. In causal language models, $\mathbf{M}_{i,j} = 0 \iff i \geq j$ ensures that no token can attend to the future.

The multi-head attention (MHA) creates multiple instances of query, key, and value matrices from an input sequence $\mathbf{X} \in \mathbb{R}^{T \times h}$ and applies the attention to each instance independently. These instances are called heads. Each head has its own mappings $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V \in \mathbb{R}^{h \times h'}$ and $\mathbf{W}_i^O \in \mathbb{R}^{h' \times h}$, where $i \in \{1..H\}$ and H is the number of heads. h' is typically set to $\frac{h}{H}$. $\mathbf{Q}_i = \mathbf{X}\mathbf{W}_i^Q, \mathbf{K}_i = \mathbf{X}\mathbf{W}_i^K, \mathbf{V}_i = \mathbf{X}\mathbf{W}_i^V$.

$$\mathbf{X}_{out} = \sum_{i=1}^{H} \text{Attention}(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i, \mathbf{M})\mathbf{W}_i^O \tag{3}$$

The resulting mechanism allows the model to adaptively focus on relevant information while maintaining differentiability. The lack of recurrence in the operations enables parallel processing of sequence elements. However, $\mathbf{Q}\mathbf{K}^\top$ is a $T \times T$ matrix and therefore introduces quadratic computational and memory complexity as a function of the sequence length.

Mixture of Experts. Mixture of Experts (MoE) combines multiple specialized neural networks (experts) with a gating mechanism that learns to route each input to the best-matching experts, activating only a small subset of experts per example. An MoE layer then computes its output as a sparsely weighted combination of the predictions of selected experts, with routing weights dynamically determined by the gating network.

Formally, given an input $\boldsymbol{x} \in \mathbb{R}^h$, the MoE layer with E experts and a scoring function (a router) $sel : \mathbb{R}^h \to \mathbb{R}^n$ can be expressed as $y(\boldsymbol{x}) = \sum_{i \in \mathcal{E}} r_i(\boldsymbol{x}) E_i(\boldsymbol{x})$ where $y(\boldsymbol{x})$ is the final output of the layer and $E_i(\boldsymbol{x})$ is the output of the expert i. $\mathcal{E}$ is the set of selected experts, usually defined as $\mathcal{E} = \text{argtopk}(r(\boldsymbol{x}) + \varepsilon, k)$, where $k \in \mathbb{N}$ is the number of active experts, ε is a stochastic noise present only during the training for exploration. The inputs are processed only by the active experts.

In contrast, Expert-Choice routing [15] ensures perfect load balancing by inverting the traditional routing paradigm. Instead of the tokens choosing their experts, the experts choose inputs they prefer to process. Given a batch of B inputs, each expert selects the top-k out of the B inputs to process.[*]

B Baselines

Apart from a dense baseline, we compare MoSA with two sparse attention methods: static, position-based sparse attention, and content-based sparse attention.

Fixed Sparse Attention. Position-based static attention patterns have been shown to be a strong sparse attention variant [6], outperforming strided sliding window attention. Fixed sparse attention

[*]In our case, each expert selects top-k tokens from the sentence to process independently for each batch.

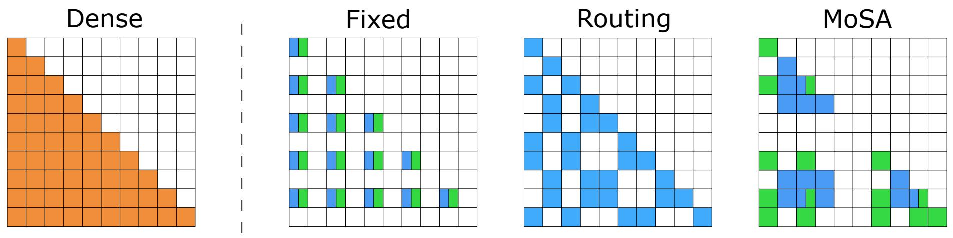

Figure 3: Attention variants visualized. In the plot, the colors indicate different heads. Sparse attention methods are roughly FLOP-matched and have sparsity $\rho = 2$. One Routing Attention head corresponds in FLOP-cost to ρ Fixed/MoSA heads. Fixed sparse attention uses only $k = \frac{T}{\rho}$ tokens in specific positions, with regular stride. The Routing Attention clusters tokens within each head into ρ clusters of size k based on their representations. MoSA selects k tokens for each attention head independently based on their representations.

for a sparsity ρ selects $k = \frac{T}{\rho}$ tokens with stride ρ. Using the notation introduced in Section 2, fixed sparse attention can be written as a special case of MoSA, where $I = [0, \rho, 2\rho, ..., T - \rho]$ and $\mathbf{r} = \mathbf{1}$.

Fixed sparse attention reduces computational complexity in two ways. First, it decreases the $O(T^2)$ cost of the full attention matrix by limiting attention to predefined token positions. Second, since only these pre-selected tokens participate in attention calculations, the query, key, value, and output transformations need only be computed for this subset rather than all tokens. This is important because MoSA also benefits from calculating transformations only for a selected subset of tokens. Hence, investigating this fixed sparse attention gives insight on whether pure benefits of sparsifying over transformations can lead to performance improvements.

However, fixed sparse attention introduces information flow constraints. Pre-selected tokens must aggregate necessary information in earlier layers. Furthermore, in the subsequent layers they have to be routed back to the positions where they are most useful. This additional overhead in information routing limits the model's representational capacity and overall expressiveness.

The Routing Transformer. We also compare MoSA to the content-based attention proposed in the Routing Transformer [11]. The Routing Attention is the most similar method to MoSA we found in the literature. It groups tokens with online K-means into ρ clusters of size $k = \frac{T}{\rho}$ *inside* each head. This is implemented during training by the top-k tokens most similar to the cluster centers using the dot-product distance metric. Cluster centers are learned using a moving average of the most similar tokens.

The Routing Attention might resemble the Expert-Choice selection with MoSA. There are, however, several crucial differences that, as our experiments show, lead to significant differences in the performance of MoSA in comparison to the Routing Transformer. Specifically, online K-means, used for clustering in the Routing Transformer is known for suffering from an extremely slow convergence rate [12]. It is also unclear if clustering keys and queries is well aligned with the language modeling objective. In contrast, the learned dynamic matching mechanism of MoSA is directly optimized by the same objective as the model.

MoSA benefits from the sparsity in the $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V, \mathbf{W}^O$ transformations, which need to be computed only for selected tokens. In contrast, the Routing Transformer has to compute all keys and queries before the clustering step. MoSA's efficiency enables the use of more heads with specialized weights in a smaller subset of tokens. Its selection can also lead to dynamic compute allocation, where some more important tokens are processed by more heads than less important tokens.

Last but not least, the Routing Transformer performs best in language modeling when the clusters share the same destination (query) tokens and source (keys and values) tokens. In our experiments, we also found that MoSA performs better if the same tokens are selected for the source and destination sides. However, to enforce this in the Routing Transformer, they require to set $\mathbf{W}^Q = \mathbf{W}^K$. In MoSA, however, the same selection for source and destination side can be enforced with $\mathbf{W}^Q$ different from $\mathbf{W}^K$, allowing greater flexibility.

We visualize typical schematic attention patterns of the baselines and MoSA in Fig. 3. Note that several previous works proposed combining different types of sparse attention to achieve synergic performance in long-sequence tasks [8, 7, 20]. In this work, we focus on investigating sparse attention methods in combination with a few dense attention heads, but without combining multiple sparse attention types. We leave combining MoSA with other sparse-attention methods for future work.

C Related Work

The quadratic cost of attention in the 2017 transformer model [1] has led to a wide body of research on efficient attention variants [21, 22]. Popular alternatives are different linear attention variants that typically use a fixed vector or matrix memory and update it recurrently. The 1992 unnormalised linear Transformers [23, 24, 25] trade performance for better computational efficiency. State space models [26, 27, 28] are popular alternatives that offer efficient, parallel training while keeping linear cost and efficient inference. The parallel training requirement forces only a linear recurrent relation between the timesteps. A common characteristic of such models is the relatively small, fixed memory that requires extreme compression. Despite recent progress, these models still underperform quadratic attention on many benchmarks [29, 30].

Sparse attention methods aim to mitigate the quadratic cost of full attention by computing attention scores for only a subset of token pairs rather than the full attention matrix. These methods typically employ various heuristics to strategically identify which tokens and token relationships are the most important to process. This is often done by introducing special tokens that serve as higher-level representations of entire chunks of tokens, or by assuming emergent hierarchical structures within the attention patterns. For example, SepLLM [31] uses separators in the sentence as special tokens that sparse attention focuses on. Sparse Transformer [6] uses static attention patterns to reduce computational complexity. Longformer [8] combines sliding window attention with additionally selected tokens globally available. BigBird [7] combines sliding window attention and global attention on selected tokens, while additionally including randomly selected tokens in the attention. Streaming LLM [32] discovers and preserves attention sinks as a necessary component despite their inefficiency and combines them with sliding window attention. Some methods [33, 34, 35] focus on post-training attention reduction, motivated by KV-cache reduction. Hash Attention[36] uses top-k selection in the attention scores to induce sparsity and improve efficiency. However, learnable sparse attention that can also be used during training [16] remains important as the quadratic cost of the self-attention mechanism is also problematic in the very costly pretaining phase.

Mixture-of-Experts (MoE) [13] have emerged as a promising paradigm for scaling model capacity without a proportional increase in computational cost. By adaptively routing input tokens to specialized experts, MoE architectures selectively activate only a part of the network. MoEs applied to transformer feedforward networks [37, 14] have been widely adapted in LLMs [38, 39, 40].

A crucial challenge in MoE is to learn a balanced routing, so that experts are utilized uniformly. Imbalanced routing leads to capacity bottlenecks when certain experts become overused while others are completely ignored. This phenomenon is called expert collapse [13]. Most approaches mitigate it by specific losses that penalize polarized expert selection [37], while others propose alternative routing methods [41, 42]. Expert-Choice routing [15] inverts the selection problem, allowing each expert to choose its preferred tokens. This way, Expert-Choice routing achieves perfect load balancing by definition, at the cost that some tokens are ignored and some are overutilized. Expert-Choice routing, however, cannot be directly applied to autoregressive modeling as it uses a non-autoregressive top-k operation over the tokens. MoD [43] proposes methods to transfer nonautoregressive expert choice routing to an autoregressive model. We leave the investigation of their adaptation to MoSA for future work.

MoE is most often applied to the feedforward part of the transformer. In contrast, some works explore MoEs in the attention mechanism to reduce the high computational cost and memory. Mixture-of-Attention Heads(MoA) [44] selects k query transformations for each token and shares a single key and value projections similarly to Multi-Query Attention(MQA) [45]. MoA allows for increasing the total number of query heads when using MQA without significantly increasing the computational cost. In contrast, MoSA selects tokens that are routed to full heads with separate queries, keys, and values (and consequently, outputs) utilizing perfect load balancing from expert choice routing for efficient sparse attention. This reduces the cost of each attention head significantly more than MoA and does not require MQA (although it might be combined for further benefits, which we leave for future work). Moreover, MoSA allows for KV-cache savings by reducing the number of selected keys, which is not possible with MoA, apart from the MQA benefit of having single KV transformations. SwitchHead [46] reduces the number of heads (and therefore the number of computed attention matrices) by adding internal experts that can compensate for the lower number of heads. This is orthogonal to MoSA and possibly can be combined for further improvements. Multi-head attention as Mixture of Head Attention [47] proposes to use dynamic weights for the output projection in order to treat the heads as experts for tokens. However, it requires calculating all attention matrices, lacking the benefits of sparse computation.

Mixture-of-Depths(MoD) [43] selects inputs to pass through a given entire transformer block to allow adaptive computation. This includes the attention mechanism. This produces efficiency gains in an FLOP-limited budget for the entire training. MoSA has multiple selection mechanisms, one for each head, and by increasing the number of heads it processes the sentence in a distributed way - each head processing its own chunk of the sentence.

D FLOPs Calculation.

Let T be the sequence length, h the hidden dimension of the model, h' the hidden dimension in each head (after passing through the query, key or value projection), k the number of tokens selected for each head, and the sparsity rate $\rho = \frac{T}{k}$.

Multiplying matrices of shape $[i, j]$ and $[j, k]$ takes precisely $(2j - 1)ik$ FLOPs. For simplicity, following common practice, we approximate it by $2jik$.

In the dense attention layer, calculating each projection (e.g., $Q_i = xW_{Q_i}$) requires $2hh'T$ FLOPs. Computing the attention matrix $QK^\top$, and multiplying the attention matrix by values V both cost $2h'T^2$ FLOPs.

Calculating the projections and attention in the MoSA head is identical, except that now we are operating on k tokens instead of T. The MoSA head involves an additional routing overhead. Calculating the routing scores costs $2hT$ FLOPs, and multiplying the intermediate values in the matrix $\in \mathbb{R}^{k\times h'}$ by the scores costs an additional $h'k$ FLOPs per head.

FLOPs cost of a single head is equal to:

$$\text{FLOP}_{\text{dense}} = \underbrace{8hh'T}_{\text{Q,K,V,O mappings}} + \underbrace{4h'T^2}_{\text{Attention}}$$

$$\text{FLOP}_{\text{mosa}} = \underbrace{8hh'k}_{\text{Q,K,V,O mappings}} + \underbrace{4h'k^2}_{\text{Attention}} + \underbrace{2hT + h'k}_{\text{routing overhead}}$$

$$\text{FLOP}_{\text{fixed}} = \underbrace{8hh'k}_{\text{Q,K,V,O mappings}} + \underbrace{4h'k^2}_{\text{Attention}}$$

$$\text{FLOP}_{\text{routing}} = \underbrace{6hh'T}_{\text{Q=K,V,O mappings}} + \underbrace{4h'k^2\rho}_{\text{Attention}} + \underbrace{2h'T}_{\text{cluster selection}} = \rho(6hh'k + 4h'k^2) + 2h'T$$

Note that, typically $k << T$, hence the MoSA head is significantly cheaper compared to a dense head.

The selection mechanism in MoSA introduces an additional overhead of $2hT + h'k$ ($2hT$ comes from token scoring and $h'k$ comes from multiplying the output by the scores), which is small compared to the rest. As a consequence, the cost of the MoSA head is comparable to that of the fixed sparsity attention head, while allowing content-based dynamic sparsity.

In contrast to MoSA and fixed attention, the Routing Transformer must compute all tokens by query, key, value, and output transformations. However, in the Routing Transformer for autoregressive text $K = Q$, therefore, only 3 projections need to be computed. Hence, the projection cost is equal to $6hh'T$. The attention in the Routing Transformer has multiple clusters inside each head. More specifically, it has ρ clusters of size k, and therefore the attention cost of the head is equal to the

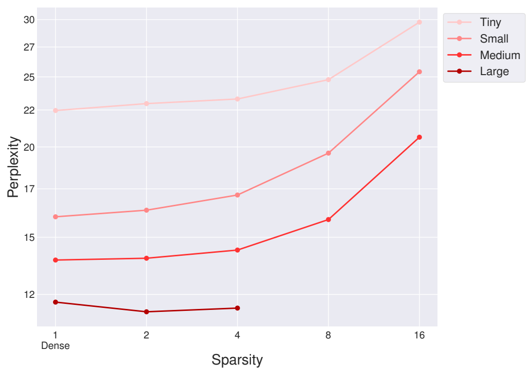

Figure 4: Perplexity of IsoFLOP matching models under pure MoSA setting. Each curve corresponds to a given FLOP budget. For a given sparsity, we replace all dense heads with a FLOP equivalent number of MoSA heads. In contrast to Fig 2, sparse models fail to outperform the baseline (apart from the Large model). This demonstrates the symbiotic relation between dense heads and MoSA heads in the hybrid model.

attention cost of the cluster multiplied by the number of clusters. The Routing Transformer has an additional layer normalization inside the head, which we omitted for simplicity.

FLOP-wise, one Routing Attention head more or less corresponds to ρ fixed attention or ρ MoSA heads. Loosely speaking, MoSA with ρ heads is similar to the Routing Attention head, where each cluster has its own custom linear transformation, rather than a single one shared among clusters.

For the multihead version, the FLOPs are multiplied by the number of heads H. There is an additional cost caused by summing the head contributions to a single output (Equations 3 and 1). However, this is already taken into account by the $2hh'TH$ cost of the output projection for multiple heads: $H(2h' - 1)hT + (H - 1)hT = (2h'H - 1)hT \approx 2hh'TH$.

Note that in the standard notation [1], the heads are first concatenated and then transformed with a single output projection instead of splitting the output operation into individual head transformations and summing. However, the result and the derivation of the FLOP counts are the same.

In the feedforward block, the intermediate layer has a typical size of $4h$. Therefore, the cost of the block is equal to $16h^2T$. Therefore, the FLOP cost of the forward pass of the entire model with l layers, a hybrid attention with H_{dense} dense heads and H_{mosa} MoSA heads is equal to:

$$lH_{dense}(8hh'T + 4h'T^2) + lH_{mosa}(8hh'k + 4h'k^2 + 2hT + h'k) + 16lh^2T$$

We omit the operations related to layer normalizations, residuals, and token embeddings from the FLOP calculations as they are negligible compared to the rest and represent an identical overhead for both dense and MoSA models. Thus, incorporating them does not influence the FLOP-matching process. This is also true for the feedforward block; yet, we still included it because it constitutes a significant portion of the total cost. We present the FLOP cost of all of our model classes (*Tiny, Small, Medium* and *Large*) in Table 4.

All models are based on the transformer architecture with Pre-layer normalisation[48]. Each model class *Tiny, Small, Medium* and *Large* follows the hyperparameters of the dense model. The necessary forward pass FLOPs are calculated according to Sec. D. The number of heads in the sparse models is set so that the resulting model is FLOP-matched to the dense baseline as closely as possible. When this is not perfectly possible, we ensure that its FLOP count never exceeds that of the baseline. For pure MoSA, all heads are replaced with MoSA heads. For the hybrid sparse models, 4 dense heads are kept, and the remaining ones are replaced with sparse heads.

E Analysing Hybrid Models

While the learned sparse attention can theoretically capture any attention pattern, the introduction of the routing mechanism complicates the learning dynamics. The router and the attention weights must be learned jointly. The router needs to identify relevant token pairs, while the attention weights

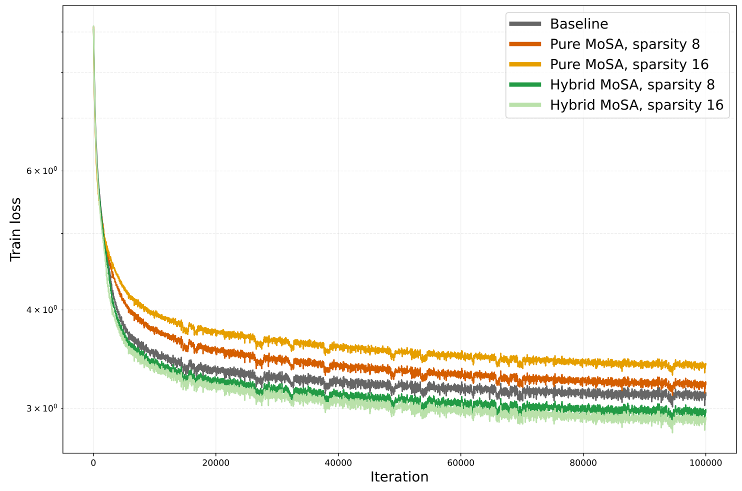

Figure 5: Training losses of the Tiny models comparing the baseline, pure MoSA, and hybrid models. The dense baseline clearly divides the models into two groups: all pure MoSA models perform worse (higher loss), while all hybrid models demonstrate superior performance (lower loss). Notably, increasing sparsity intensifies the difference for both model types: hybrid models achieve progressively lower loss with greater sparsity, whereas pure MoSA models show increasingly higher loss as sparsity increases. Additionally, the early training phase (between 5,000 and 10,000 steps) reveals a distinct pattern where pure MoSA models experience a more rapid slowdown in their learning progress compared to both dense and hybrid models.

learn to process these selected interactions. This interdependence can lead to training instabilities, particularly in the early stages, when router decisions are largely random. Poor initial routing can prevent attention heads from learning meaningful patterns, while the lack of meaningful patterns prevents the router from learning to select important tokens, creating a vicious circle.

Our preliminary experiments have shown that pure MoSA models without additional dense heads fail to improve the perplexity of dense baselines. To verify this, we conducted a study similar to our main results in Sec. 3. We gradually increase the sparsity by replacing all dense heads with MoSA heads while maintaining an identical FLOP count to the baseline. We do this by finding the maximum number of MoSA heads for which the FLOP count remains lower than the baseline. The results, shown in Fig. 4, demonstrate that increasing sparsity monotonically worsens model performance in most settings. This performance degradation with pure MoSA heads likely stems from the stability issues explained in the previous paragraph.

Interestingly, the largest model is an exception, and initially there is a visible improvement from 12.20 baseline perplexity to 11.83 perplexity of the FLOP-matched pure MoSA model with sparsity 2. This is still significantly worse than the 11.15 perplexity of the hybrid model with sparsity 4. Moreover, the saturation is much faster than for hybrid models. For hybrid models, the sparsity around 32 or 64 seems to be optimal. In contrast, for the MoSA-only model, the best perplexity is reached for sparsity 2 for the *Large* budget and 1 for the smaller ones. However, the conclusion is consistent across all scales: hybrid MoSA models significantly outperform MoSA-only models, which generally underperform the dense baseline. Thus, hybridization seems necessary.

The impact of sparsification is also visible in the training characteristics. Compared to the baseline, pure MoSA models start to plateau faster. While the losses of dense and hybrid models continue to show steep initial improvement, pure MoSA models slow down much sooner. This supports our hypothesis about the difficulty of learning the routing and attention simultaneously. We compare the training losses in Fig. 5.

Optimal Number of Dense Heads Hybrid models consistently outperform pure MoSA models. This raises a natural question: What is the optimal ratio of dense to sparse heads and how does this ratio relate to the sparsity rate?

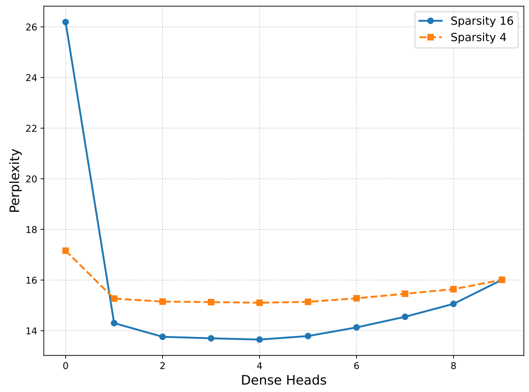

Figure 6: Perplexity of the FLOP matched models with a different number of dense heads for sparsities 4 and 16. 9 dense heads correspond to the dense baseline.

To answer these questions, we conducted a series of experiments in which we varied both the sparsity factor of MoSA heads and the number of dense heads while keeping the total FLOP budget constant. We choose to use the *small* model and investigate sparsities $\rho = 4$ and $\rho = 16$, while we set the number of dense heads in the hybrid model from 0 to 9 (full dense model) and adapt the number of sparse heads to match the FLOP budget. Our results are shown in Fig. 6. We can see that the optimal number of dense heads in this case is 4 and is sparsity-agnostic. Because of this, we chose to use 4 dense heads in our main experiments in Sect. 3. Furthermore, we observe that it is critical to have at least one dense head. Having more than one has diminishing returns, and having more than 4 has a negative effect on the performance. The plot also shows that the lack of dense heads is more hurtful for models with higher sparsities. We conclude that in our case, 4 heads are sufficient to stabilize the training, and it is better to allocate the remaining FLOP budget to the more efficient MoSA heads.

F Resource Optimization

The previous section demonstrates MoSA's ability to achieve better perplexity than dense transformers with an identical compute budget. In this section, we examine MoSA's practical efficiency gains. Specifically, we match the perplexity scores between the MoSA and the dense baseline to measure wall-clock time, memory, and KV-cache size savings.

To find the perplexity-matched comparison, we select sparsity to be equal to 32 for model sizes *Tiny, Small* and *Medium*. For *Large* we select $\rho = 16$ to keep sparsity closer to the range investigated in Section 3. Then, we gradually increase the number of MoSA heads until the perplexity matches the dense baseline. We do it for all four model scales defined in Section 3.

The results are shown in Table 1. MoSA can match the dense baseline, while being faster in wall-clock time and using less memory at the same time. These findings show that MoSA not only improves model quality in the FLOP-matched setting but can also be used to reduce computational and memory requirements when targeting the same performance level. Furthermore, it shows that MoSA uses computation more effectively than standard dense attention across all efficiency metrics. MoSA achieves this without a specialized CUDA kernel using only PyTorch-level operations. We expect that designing a specialized kernel would result in additional significant efficiency gains.

In addition to the speed and memory used for the training, we report the total number of key-value pairs (KV) used, calculated as KV $= TH_{dense} + kH_{mosa}$, where H_{dense} and H_{mosa} represent the number of dense and sparse heads, respectively. KV directly corresponds to the size of the costly KV-Cache in the autoregressive setting. KV cache optimization has been the goal of many post-training sparse-attention methods[33, 34, 35]. Our results demonstrate that MoSA offers a significant reduction in KV-cache size while simultaneously improving speed and memory requirements.

	Tiny		Small		Medium		Large	
	Dense	MoSA	Dense	MoSA	Dense	MoSA	Dense	MoSA
Dense Heads	9	4	9	4	9	4	16	4
MoSA Heads	0	17	0	14	0	12	0	16
Perplexity (↓)	22.46	22.40	16.02	16.01	13.94	13.76	12.20	12.16
Wall-time/step ↓ (ms)	137	**127**	326	**319**	619	**592**	807	**703**
Wall-time/step gain (%)	–	-7.3%	–	-2.1%	–	-4.4%	–	-12.9%
Memory ↓ (GB)	21.1	**19.0**	32.4	**31.4**	50.2	**49.4**	104.1	**94.5**
Memory gain (%)	–	-10.0%	–	-3.1%	–	-1.6%	–	-9.2%
KV Total ↓ (K)	9.2	**4.5**	9.2	**4.4**	9.2	**4.4**	16.4	**5.0**
KV Total gain (%)	–	-51.1%	–	-52.2%	–	-52.2%	–	-69.5%

Table 1: Resource usage reduction from perplexity-matched MoSA models. KV is the KV-cache size, representing the total number of key-value pairs required (in thousands). MoSA models match the perplexity of dense baselines while at the same time improving wall-clock time, using less memory, and significantly smaller KV cache for all model sizes. Resource usage was measured on a single A100 GPU for *Tiny, Small* and *Medium* models and on two A100 GPUs for *Large*.

G Scaling with Sequence Length

Traditionally, sparse attention methods have been introduced as a necessity when sequence length makes dense attention computationally prohibitive. After demonstrating MoSA's effectiveness in standard-length sequences, we now investigate MoSA's benefits in this long sequence setup.

In contrast to previous sections, here we combine MoSA or a baseline method with local attention [6, 8]. We use local attention instead of dense attention because even a small number of dense attention heads would result in prohibitive memory usage in a longer context scenario. This is a standard practice in the sparse attention literature [6, 11]. Local attention preserves local dependencies, while global, sparse attention enables efficient processing of long dependencies.

We scale our sequence length from 1024 to 8192 tokens and keep the k constant equal to 64. Hence, the sparsity increases from $\rho = 16$ for $T = 1024$ to $\rho = 128$ for $T = 8192$. Contemporary sparse attention methods for long sequences are trained in longer sequences [16]. However, due to our limited hardware budget, we restrict our experiments to a sequence length of 8192. We treat this investigation as a preliminary analysis that demonstrates the potential of MoSA for long sequences. Importantly, it demonstrates that MoSA performs well when combined with local attention, which is a typical long-sequence setup.

As in the previous section, we compare MoSA with fixed sparse attention and the Routing Attention. All long sequence models have 6 layers and hidden dimension size of 1024. The Routing Transformer has 4 local attention heads and 4 Routing Transformer heads in all layers, whereas the fixed sparse attention and MoSA have 60 sparse heads and 4 local attention heads. We chose 60 sparse heads to roughly FLOP match all models for $T = 1024$. However, as we keep k constant, for longer sequences with $2048, 4096$ and 8192 tokens, the FLOP cost for fixed attention and MoSA will be much lower than for the Routing Attention. For $T = 8192$ FLOP cost of 60 MoSA's heads is equal to only 22.99% of 4 Routing Transformer heads.

The results are shown in Fig. 7. MoSA significantly outperforms other sparse attention methods across all sequence lengths. This is true even at length 8192, where MoSA uses only a small fraction of the computational cost of the Routing Transformer. The significant performance gap in the results demonstrates the potential of MoSA for ultra-long sequences [21, 16, 49]. Given our limited resources, we leave the investigation of MoSA in this context for future work.

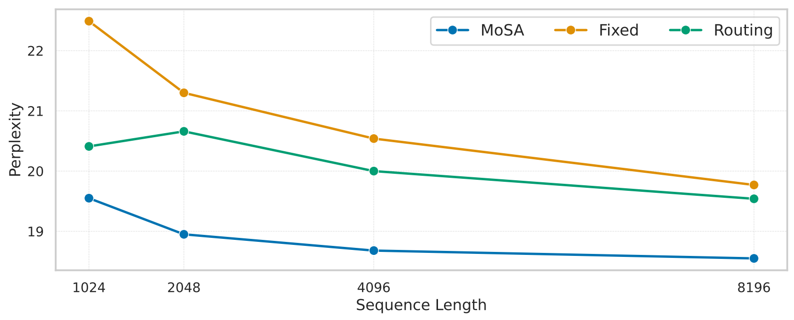

Figure 7: Perplexity of sparse-attention methods (MoSA, Fixed, and Routing) as sequence length increases. Each method has a fixed size window size (cluster size for the Routing Transformer, number of tokens selected for each head in MoSA and Fixed) regardless of total sequence length. MoSA matches the computational cost of the fixed sparsity baseline while requiring fewer FLOPs than the Routing Attention and consistently achieves the lowest perplexity.

H Downstream Tasks

We evaluate the zero-shot downstream performance of MoSA on six established benchmarks: LAMBADA [50], WinoGrande [51], BLiMP [52], HellaSwag [53], PIQA [54] and AI2ARC [55]—covering tasks from cloze-style completion to commonsense reasoning.

During training, MoSA operates on sequences of more or less constant size $T = 1024$. However, for downstream tasks, some inputs will be much shorter. For example, most datapoints in the BLiMP dataset do not exceed 10 tokens. In order to handle such situations, we adaptively choose the number of tokens for each input to be $k = \max(\lfloor \frac{T}{\rho} \rfloor, 2)$ tokens for each head. This simulates the ratio of tokens selected for the attention head during the training. Moreover, it ensures that at least 2 tokens are selected, which is the minimum necessary for the attention to model any cross-token dependencies.

For each scale and sparse model type, we select the model with sparsity $\rho > 1$ that produced the best perplexity in the IsoFLOP scenario (Sec. 3). We also include the dense baseline for each size. Table 2 reports the performance across the tasks. The best result for a given task across model types is bold.

For *Tiny*, *Small*, and *Medium* scales, MoSA generally outperforms other models. BLiMP stands as a notable exception, where MoSA consistently underperforms. This weak performance on BLiMP can be attributed to the extremely short length of most examples in the dataset. With longer sequences seen during training, each MoSA head can selectively process only the tokens it handles well. However, in short sequences, the shortage of tokens forces MoSA heads to operate on tokens outside their training distribution. Furthermore, when $\lfloor \frac{T}{\rho} \rfloor = 1$, resulting in only 2 tokens being selected, there is a significant discrepancy between the percentage of selected tokens compared to training conditions. Models with a high sparsity factor of 64 typically select only 1.56% tokens in a sequence for each attention head. Yet for a sequence length of $T = 10$, 2 selected tokens represent 20% of the sentence, creating a distribution mismatch.

Moreover, in *Large* scale, the Dense baseline outperforms MoSA despite having much higher perplexity. We attribute the downstream performance gap of MoSA to two main factors. First, MoE architectures have been shown to suffer from expert overspecialization, which often leads to decreased performance in downstream tasks [14, 56]. Instruction tuning has been shown to mitigate this issue [57].

Furthermore, content-based sparse attention methods tend to struggle on shorter sequence[†]. Our experiments confirm this pattern, as MoSA outperforms the Routing Attention in most tasks. Furthermore, some runs of the Routing Attention were unstable in context of downstream tasks (Medium scale of the Routing Attention). Practitioners report that extending training by additional epochs on truncated sequences can mitigate the issues of sparse attention methods on short sequences[†].

[†]See: https://github.com/lucidrains/routing-transformer?tab=readme-ov-file#issues

	Model	LAMBADA	WinoGrande	BLiMP	HellaSwag	PIQA	AI2ARC
Tiny	Dense	18.7	50.3	72.0	27.5	**59.4**	28.0
	MoSA	**26.5**	**53.0**	65.5	**29.1**	**59.7**	**29.4**
	Routing	14.0	51.3	66.2	27.8	57.1	25.9
	Fixed	17.1	50.6	**72.5**	27.7	58.6	28.1
Small	Dense	25.8	**52.1**	**76.2**	30.9	62.4	30.1
	MoSA	**29.4**	51.9	70.5	**31.9**	**63.2**	30.0
	Routing	19.2	50.7	70.2	28.0	57.6	27.3
	Fixed	24.6	51.6	75.3	30.1	**63.2**	**30.2**
Medium	Dense	31.4	51.2	**77.8**	33.8	64.5	**31.5**
	MoSA	**36.1**	**52.1**	66.1	**34.2**	63.3	31.4
	Routing	10.2	51.5	65.9	30.3	57.8	27.8
	Fixed	29.4	51.4	77.3	33.0	**64.6**	**31.5**
Large	Dense	**36.2**	**52.5**	**80.4**	**38.7**	**67.1**	**33.8**
	MoSA	35.0	51.4	74.2	37.5	65.9	31.7
	Routing	27.5	51.1	76.5	36.2	64.1	32.5
	Fixed	32.3	51.7	79.6	35.9	66.0	32.2

Table 2: Accuracy on downstream zero-shot tasks. Each model is selected with the best sparsity in the IsoFLOP comparison. Note that on downstream tasks, the token selection mechanism of MoSA operates out of distribution. Despite this, MoSA often outperforms the dense baseline. Even when it doesn't, the performance gap is usually small.

I Limitations and Future work

Due to the top-k selection over tokens, MoSA is non-autoregressive in nature and requires adaptations to be directly applicable to the autoregressive scenario. This is true not only for MoSA, but for all expert-choice routing methods, as well as for the Routing Transformer that uses non-autoregressive clustering. MoD proposed to solve this problem by learning an autoregressive classifier post-training to predict if the given token would have been selected by the non-autoregressive router or not. We consider exploring this issue in depth as an important future direction.

The perplexity gains do not always translate to downstream task performance (App. H). This discrepancy stems from two distinct factors: First, sparse attention methods generally underperform on tasks consisting of short sequence lengths. Practitioners have shown that additional training with truncated sequences might alleviate this problem. Second, MoE architectures experience performance gaps in downstream tasks despite strong language modeling capabilities, although recent research demonstrates that instruction tuning can help significantly [57]. We consider exploring methods to mitigate the discrepancy between perplexity and downstream task performance in future work.

Several promising research directions emerge from this work. Further exploration of MoSA's effectiveness on longer sequences remains an important direction. Furthermore, combining multiple sparse attention methods often leads to synergic improvements on long sequences [7, 8]. Thus, we expect that combining other sparse head types with MoSA could lead to additional benefits.

From an implementation perspective, developing specialized CUDA kernels would further improve efficiency. MoSA could be integrated with complementary approaches such as MQA[45], GQA[58], or SwitchHead[46] to improve the efficiency even further. Furthermore, exploring MoSA on other modalities, particularly vision transformers, could yield valuable insights into the method's versatility across different data types and architectures.

Model size	#Params Dense	Dense ppl ↓	MoSA Best ppl ↓	Fixed Best ppl ↓	Routing Best ppl ↓
Tiny	28M	22.46	16.37 (-27.1%)	23.28 ($+3.7\%$)	23.33 ($+3.9\%$)
Small	113M	16.01	12.97 (-19.0%)	16.51 ($+3.1\%$)	16.43 ($+2.6\%$)
Medium	210M	13.95	11.22 (-19.6%)	14.35 ($+2.9\%$)	14.21 ($+1.9\%$)
Large	516M	12.20	11.15 (-8.6%)	12.40 ($+1.6\%$)	12.24 ($+0.3\%$)

Table 3: Comparing dense and sparse models (Fixed, Routing, MoSA) under a fixed computational budget (see Section 3). For sparse models, the table contains the best perplexity across all sparsities bigger than 1. The results for sparse models were selected as the best of all sparsities. Relative difference to the dense baseline is displayed in the parentheses. MoSA significantly outperforms the dense baseline, reducing perplexity by up to 27%. The fixed and the Routing Transformer baselines both fail to reach the performance of the dense model.

J Details of the Models

We use four model sizes for our experiments: *Tiny*, *Small*, *Medium* and *Large*. Each size is defined by the *FLOP count* of the forward pass of the corresponding dense transformer baseline. The parameter count of dense models associated with each size is: 28M for *Tiny*, 113M for *Small*, 210M for *Medium*, and 516M for *Large*.

In the Table 4 we list hyperparameters all of dense baselines.

Implementation details We use the SentencePiece [59] tokenizer based on sub-word units [60, 61] a vocabulary size of 8000. All our models are trained on the C4 [62] dataset for 100k batches, with batch size $B = 64$ and sequence length $T = 1024$. This means that we train on the $10^5 SB \approx 6.5B$ tokens from the dataset. We use the Adam [63] optimizer with a learning rate of 0.00025, gradient clipping above the norm of 0.25, and a linear warmup for 4k steps. For detailed hyperparameters, please refer to Appendix J.

Positional encodings. All our experiments use Rotary Positional Encodings (RoPE) [64]. RoPE applies positional encodings for each attention head after query and key mapping. It does this by rotating them at an angle determined by the token's position in a sentence. Similarly to the attention mask, we must ensure that the rotations correspond to the token's original position in the sequence $\mathbf{X}$ rather than the selected subset $\boldsymbol{X}^S$. Thus, we adapt RoPE to be aware of token positions I. Following standard practice, we rotate half of the dimensions and leave the other half unchanged.

Resources Used. All the experiments in the paper were run on NVIDIA-A100 80GB nodes with GPUs ranging from 1 for small experiments to 4 GPUs. Each experiment was limited by 24h operation time.

	Tiny	**Small**	**Medium**	**Large**
FLOPs per pass (G)	54.76	219.85	430.70	1,130.65
Layers	6	9	18	27
Hidden size	512	1,024	1,024	1,280
Feedforward hidden size	2,048	4,096	4,096	5,120
Head hidden size	64	64	64	64
Number of heads	9	9	9	16

Table 4: Hyperparameters of the different model variants and the corresponding FLOP cost of the forward pass for a sequence length of $T = 1024$.

		Sparsity								
		1	2	4	8	16	32	64	128	256
		Perplexity (↓) for given sparsity								
Tiny	MoSA	22.46	21.76	20.45	19.24	18.00	16.90	**16.37**	17.27	18.06
	Pure MoSA	**22.46**	22.96	23.30	24.78	29.76	-	-	-	-
Small	MoSA	16.01	15.69	15.10	14.33	13.68	**12.97**	13.30	-	-
	Pure MoSA	**16.01**	16.35	17.16	19.61	25.41	-	-	-	-
Med.	MoSA	13.95	13.52	12.81	12.16	11.47	**11.22**	-	-	-
	Pure MoSA	**13.95**	14.03	14.40	15.87	20.63	-	-	-	-
Large	MoSA	12.20	11.67	**11.15**	-	-	-	-	-	-
	Pure MoSA	12.20	**11.83**	11.97	-	-	-	-	-	-
		Number of parameters for given sparsity								
Tiny	MoSA	28M	34M	48M	78M	136M	242M	423M	693M	1B
	Pure MoSA	28M	39M	65M	119M	222M	-	-	-	-
Small	MoSA	113M	127M	163M	229M	360M	599M	1B	-	-
	Pure MoSA	113M	142M	203M	324M	559M	-	-	-	-
Med.	MoSA	210M	239M	310M	442M	703M	1.2B	-	-	-
	Pure MoSA	210M	267M	390M	632M	1.1B	-	-	-	-
Large	MoSA	516M	650M	943M	-	-	-	-	-	-
	Pure MoSA	516M	703M	1B	-	-	-	-	-	-
		Number of MoSA heads for given sparsity								
Tiny	MoSA	0	13	31	69	142	276	505	848	1277
	Pure MoSA	0	23	56	124	255	-	-	-	-
Small	MoSA	0	11	26	54	109	210	381	-	-
	Pure MoSA	0	21	47	98	197	-	-	-	-
Med.	MoSA	0	11	26	54	109	210	-	-	-
	Pure MoSA	0	21	47	98	197	-	-	-	-
Large	MoSA	0	27	60	-	-	-	-	-	-
	Pure MoSA	0	37	80	-	-	-	-	-	-

Table 5: Detailed statistics of the main IsoFLOP experiments from Sec. 3. Models Tiny, Small, Medium, and Large are as described in App.J. Sparsity 1 corresponds to dense baselines. Pure MoSA models for sparsities ≥ 1 have only MoSA heads, calculated as the biggest number of heads that will not increase the FLOP budget of the dense baseline (other hyperparameters stay the same as in the baseline). MoSA models have 4 dense heads and the rest of the heads are sparse, calculated such that the flop cost of both dense and sparse heads is lower than the baseline. Therefore, the total number of heads in hybrid models (with sparsity ≥ 1) is the number shown in the bottom table + 4. For perplexity, the best result for each row is bold.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims are justified quantitatively in the Section 3

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations in the Section I.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: The paper does not include any theoretical results. The calculation of the FLOPs that is in the paper is detailed in the Appendix D.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We describe the models used in the experimental section and specific details of all models in the Appendix J.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The data we used is public, we will include the code necessary to reproduce the results as a supplementary material

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We specify the experimental and implementation details in the Section 3 and Appendix J.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Each experiment corresponds to training a language model from scratch, making it extremely expensive to create error bars. Furthermore, the consistency of MoSA improvements across different sparsities together with the level of improvement strongly suggest that the improvement is not by random chance.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We discuss the resources used in the experiments in App.J and in Sec. F.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We briefly discuss the broader impact in the Conclusions.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our released models are too small to pose a risk of misuse that could not be done with other publicly available models.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the datasets used for the models. In the included code we also list explicitly the library dependencies.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Together with the code we release a documentation on how to recreate main results.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.