Exploiting Activation Sparsity with Dense to Dynamic-k Mixture-of-Experts Conversion

Filip Szatkowski [*] [1] [2] **Bartosz Wójcik** [*] [2] [3] **Mikołaj Piórczyński** [*] [1] **Simone Scardapane** [4]

Abstract

Transformer models can face practical limitations due to their high computational requirements. At the same time, they exhibit high activation sparsity, which can be leveraged to reduce the inference cost by converting parts of the network into equivalent Mixture-of-Experts (MoE) layers. Despite the crucial role played by activation sparsity, its impact on this process remains unexplored. In particular, we show that the efficiency of the conversion can be significantly enhanced by a proper regularization of the activation sparsity of the base model. Moreover, motivated by the high variance of the number of activated neurons for different inputs, we introduce a more effective dynamic-k expert selection rule that adjusts the number of executed experts on a per-token basis. The proposed method, Dense to Dynamic-k Mixture-of-Experts (D2DMoE), outperforms existing approaches on common NLP and vision tasks, allowing us to save up to 60% of inference cost without significantly affecting model performance.

1. Introduction

Transformers have become a predominant model architecture in various domains of deep learning such as machine translation (Vaswani et al., 2017), language modeling (Devlin et al., 2018; Radford et al., 2019), and computer vision (Dosovitskiy et al., 2020; Khan et al., 2022). The effectiveness of Transformer models in various applications is closely related to their ability to scale efficiently with the number of model parameters (Kaplan et al., 2020), prompting researchers to train progressively larger and larger models (Touvron et al., 2023; Jiang et al., 2023). However, the considerable computational demands of these models often restrict their deployment in settings with limited resources.

At the same time, Transformer models exhibit considerable activation sparsity (Li et al., 2022), which suggests that most of their computations are redundant. Conditional computation methods can reduce these unnecessary costs by using only a subset of the model parameters for any given input (Han et al., 2021). In particular, Mixture-of-Experts (MoE) layers (Shazeer et al., 2016) consisting of multiple sparsely executed experts are an effective way to decouple the number of parameters of the model from its computational cost (Clark et al., 2022). As shown by (Zhang et al., 2022), many pre-trained dense Transformer models can be made more efficient by converting their FFN blocks into MoE layers, a process they call *MoEfication*.

Contributions of this paper: We consider the following research question: what is the *optimal* way to convert a generic Transformer model into an equivalent sparse variant? We propose to improve upon the MoEfication process with Dense to Dynamic-k Mixture-of-Experts (D2DMoE) and demonstrate it in Figure 1. Our findings can be summarized as follows.

1. We analyze the relationship between the activation sparsity of the starting model and the efficiency of the converted MoE model. We show that computational savings are directly related to sparsity levels, and we correspondingly enforce higher activation sparsity levels before conversion through a lightweight fine-tuning process, improving the cost-to-performance trade-off.

2. We identify the router training scheme in the original MoEfication algorithm as a limitation of the conversion process and propose to instead frame the router training as a regression problem so that our routers directly predict the norm of the output of each expert.

3. We show that Transformer models exhibit significant variance in the number of activated neurons, and standard top-k expert selection in the MoE layers is inefficient. We propose an alternative dynamic-k expert selection scheme that adjusts the number of activated experts on a per-token basis, further increasing the

[*]Equal contribution [1]Warsaw University of Technology [2]IDEAS NCBR [3]Jagiellonian University [4]Sapienza University of Rome. Correspondence to: Filip Szatkowski <filip.szatkowski.dokt@pw.edu.pl>.

Efficient Systems for Foundation Models Workshop at the 41st International Conference on Machine Learning, Vienna, Austria. Copyright 2024 by the author(s).

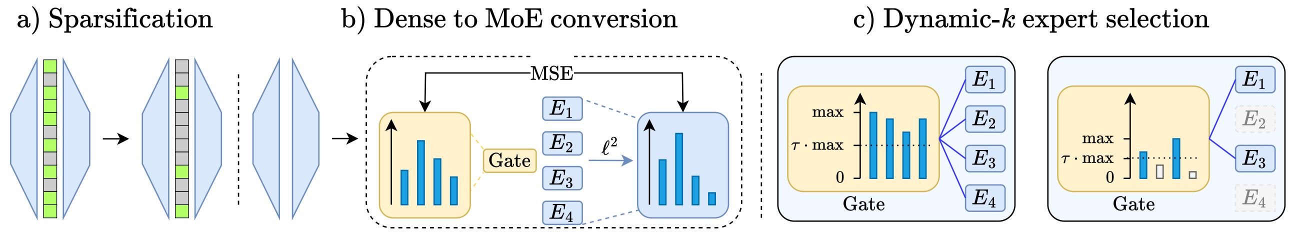


Figure 1: Key components of D2DMoE: (a) We enhance the activation sparsity in the base model. (b) We convert FFN layers in the model to MoE layers with routers that predict the contribution of each expert. (c) We introduce dynamic-k routing that selects the experts for execution based on their predicted contribution.

overall efficiency.

We evaluate our method on image classification and language modeling, and show that in all cases it achieves significant improvements upon the baseline in terms of cost-vs-performance trade-offs.

2. Motivation

MoE models have gained a lot of traction over the last years as an effective architecture to decouple the parameter count from the computational cost of the models (Zoph et al., 2022). In a MoE layer, hard sparsity is usually enforced *explicitly* by applying a top-k operation on the outputs of a trainable gating layer. However, many recent works (Zhang et al., 2023; Chen et al., 2023; Qiu et al., 2024) have shown that most Transformers, when trained at scale, build *intrinsically* sparse and modular representations. Zhang et al. (2022) proposed to leverage this naturally emerging modularity with MoEfication - a method that converts dense transformer models into MoE models by grouping FFN weights into experts and subsequently learning small routers that determine which experts to activate. Models converted with MoEfication are able to preserve the performance of the original dense models while using only a fraction of their computational cost. However, we believe that the MoEfication procedure is not optimal, and therefore aim to obtain dense-to-sparse conversion schemes that obtain a better cost-performance trade-off.

Intuitively, a MoE converted from a sparser base model would be able to perform the original function using a smaller number of experts. To validate this hypothesis, we perform MoEfication on different variants of GPT2-base with varying activation sparsity levels and show the results in Figure 2a. As expected, MoEfication performs better with sparser models. We further investigate the per-token mean and the variance of non-zero neurons in the base and sparsified model. As visible in Figure 2b, different layers use a different number of neurons on average. Moreover, the variance of the number of activated neurons is quite high and becomes even more significant in the sparsified model. This means that static top-k gating as used in MoEfication is not optimal for dense-to-MoE converted models, and a more flexible expert assignment rule that would be able to handle the high per-token and per-layer variance could be beneficial to the efficiency of such models, as illustrated at Figure 2c. Such dynamic-k gating requires routers that reliably predict the contribution of each expert. We observe that routers obtained through MoEfication do not accurately reflect this contribution. Moreover, their router training procedure depends on the strict sparsity of the model guaranteed by the ReLU activation function. Therefore, we design a novel router training scheme that directly predicts the contribution of each expert and generalizes to the broader family of activation functions. We combine the proposed components (sparsity enforcement, expert contribution routing, and dynamic-k gating) into a single method that we call Dense to Dynamic-k Mixture-of-Experts (D2DMoE), which we describe in detail in the next Section.

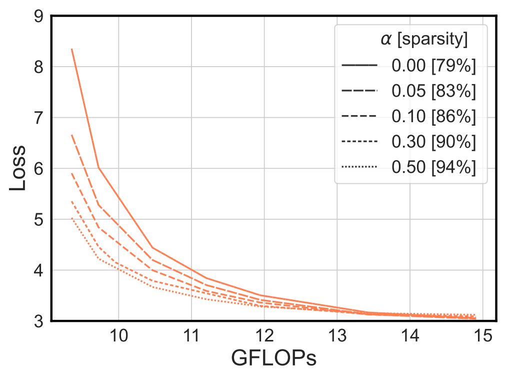
3. Method

D2DMoE reduces the computational cost of the model by splitting every MLP module into a MoE layer. In this section, we describe all of its components in detail. A high-level overview of the entire procedure is presented in Figure 1.

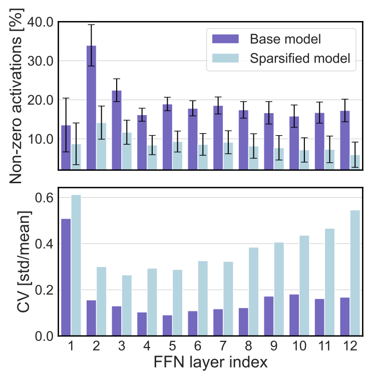
3.1. Enforcing activation sparsity

We expect that enforcing higher levels of activation sparsity may allow for the execution of an even smaller number of experts, resulting in overall computational savings. To this end, we induce activation sparsity by fine-tuning the model with an additional loss term that induces activation sparsity (Georgiadis, 2019). We apply the *square Hoyer* regularization (Kurtz et al., 2020; Hoyer, 2004) on the activations of the model:

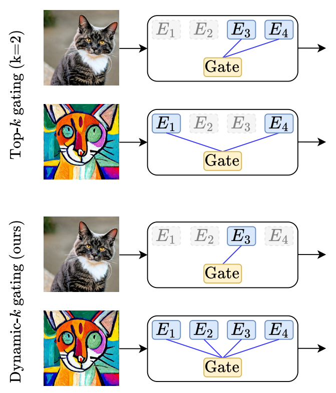
$$\mathcal{L}_s(x) = \frac{1}{L} \sum_{l=1}^{L} \frac{(\sum_i |a_i^l|)^2}{\sum_i {a_i^l}^2}, \tag{1}$$



(a) Impact of sparsity on MoE conversion



(b) Non-zero activations distribution



(c) Top-k vs dynamic-k gating

where a^l is the activation vector from the middle layer of the l-th MLP for input x, and L is the total number of MLPs in the model. Overall, the model is trained with the following cost function:

$$\mathcal{L}(x) = \mathcal{L}_{\text{CE}}(\hat{y}, y) + \alpha \mathcal{L}_s(x) \tag{2}$$

where $\mathcal{L}_{\text{CE}}$ is cross-entropy loss, and α is the hyperparameter that controls the strength of sparsity enforcement. We find that the pre-trained models recover the original performance with only a fraction of the original training budget.

3.2. Expert clustering

We split the two-layer MLP modules into experts using the parameter clustering method proposed by Zhang et al. (2022). Assuming the MLP layers are composed of weights $\boldsymbol{W}_1$, $\boldsymbol{W}_2$ and corresponding biases $\boldsymbol{b}_1$, $\boldsymbol{b}_2$, we treat the weights of each neuron from $\boldsymbol{W}_1$ as features and feed them into the balanced k-means algorithm (Malinen and Fränti, 2014) that groups neurons with similar weights together. Then, we use the resulting cluster indices to split the first linear layer $\boldsymbol{W}_1$, the first bias vector $\boldsymbol{b}_1$, and the second linear layer $\boldsymbol{W}_2$ into n experts of the same size. The second bias $\boldsymbol{b}_2$ is not affected by this procedure.

MoEfication process was designed for standard two-layered MLPs (Zhang et al., 2022). Recent LLMs (Touvron et al., 2023; Team, 2024) have shifted towards *gated* FFNs, where the activation is realized through a Gated Linear Unit (GLU) (Shazeer, 2020), which contains an additional weight matrix for the gate projections. To adapt the expert clustering procedure described above to gated FFN layers, we cluster the weights of the gating matrix $\boldsymbol{W_g}$ instead of $\boldsymbol{W_1}$, and use the obtained indices to divide the weights of the two other layers. We provide more intuition and details on our method for gated FFNs in Appendix E.

3.3. Expert contribution routing

In a standard MoE-based model, the gating networks are trained in an end-to-end manner. Contrary to this, we train each gating network independently. We propose to frame the problem of training the router as a regression task and directly predict the ℓ^2-norm of the output of each expert with the router. Formally, given an input token z, the D2DMoE router R is trained to minimize the following loss:

$$\mathcal{L}_r(z) = \frac{1}{n}\sum_{i}^{n}\left(R(z)_i - \|E_i(z)\|\right)^2 \tag{3}$$

where E_i is the i-th expert. We use a small two-layer neural network as the router R and apply an absolute value activation function to ensure non-negative output. This regression-based formulation is still compatible with the commonly used top-k expert selection, but enables more precise attribution of the contribution of each expert, as we show later in the experimental section.

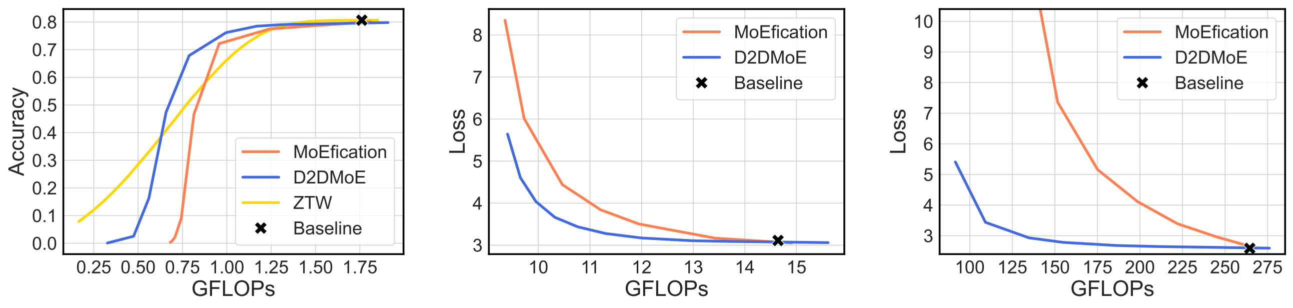
Note that Zhang et al. (2022) also trains each routing network independently, but their method constructs artificial labels for each input, and then subsequently trains the router as a classifier. We discuss the differences in Appendix C.

3.4. Dynamic-k gating

Commonly used MoE layers always execute top-k experts for each token, where k is a predefined hyperparameter. This means that, regardless of the difficulty of the input, the model spends the same amount of compute on each batch (Zhou et al., 2022) or token (Shazeer et al., 2016). While this may be appropriate if the model is trained with the same restriction, it is suboptimal for a model that was converted from a dense model, as we show in Section 2.

Since our router directly predicts the ℓ^2-norm of output of each expert, we propose a dynamic-k expert selection method that skips experts for whom the router predicts relatively small output norms. Given a router output vector $R(z)$, we select a hyperparameter $\tau \in [0, 1]$ and define the expert selection rule G for the i-th element as:

$$G(z)_i = \begin{cases} 1 & \text{if } R(z)_i \geq \tau \cdot \max R(z) \\ 0 & \text{if } R(z)_i < \tau \cdot \max R(z) \end{cases} \tag{4}$$



(a) ViT-B on ImageNet-1k. (b) GPT-2-base on OpenWebText (c) Gemma-2B on C4

Figure 3: FLOPs vs loss comparison for our method and MoEfication (Zhang et al., 2022) on vision and NLP benchmarks. Our method outperforms the baselines at every computational budget.

Note that as τ increases, the number of executed experts decreases along with the overall computational cost. We emphasize that after model deployment τ can be adjusted without the need for retraining.

4. Experiments

To analyze the impact of our method, we evaluate its performance on image classification and language modeling. We obtain performance versus computational cost characteristics for each method by evaluating the methods with different inference hyperparameters (either τ described in Section 3.4 for D2DMoE or number of experts k for MoEfication). We report the computational cost of each method in FLOPs, as it is a device-independent metric that has been shown to correlate well with latency (Mirzadeh et al., 2023).

For MoEfication, we follow the procedure described by Zhang et al. (2022) by converting the activation functions of the pre-trained model to ReLU and then fine-tuning the model. In the case of D2DMoE, we also replace activation functions with ReLU. To provide a fair comparison, the total training data budget is always for all methods. See Appendix G for a detailed description of our setup.

Image classification Vision Transfomers (Dosovitskiy et al., 2020) are becoming one of the most popular architectures in computer vision. Since our method applies to any Transformer model, we evaluate D2DMoE on the popular ImageNet-1k (Russakovsky et al., 2015) dataset. We use a pre-trained ViT-B checkpoint as the base model and compare D2DMoE with MoEfication in terms of the computational cost versus accuracy trade-off. For broader comparison, we also evaluate the state-of-the-art early-exit method Zero-time Waste (ZTW) (Wójcik et al., 2023). Our results, presented in Figure 3a, demonstrate the significant gains from applying our method over MoEfication.

Language modeling We evaluate our method on language modeling and compare it with MoEfication using GPT-2-base (Radford et al., 2019) and Gemma-2B (Team, 2024). We initialize GPT-2 models from a publicly available OpenAI checkpoint pre-trained on a closed-source WebText dataset, and use OpenWebText (Gokaslan and Cohen, 2019) in all of our experiments. For Gemma-2B, we also start from the publicly available pretrained model and evaluate its language capabilities on the C4 dataset (Raffel et al., 2020) after finetuning. For both models, we use around 1B tokens for the finetuning phase (less than 1% of the cost of original pretraining) and 8-16M tokens for router training.

We present test losses for D2DMoE and MoEfication at different compute budgets for GPT-2-base and Gemma-2B in Figures 3b and 3c respectively. Our method outperforms the baseline at every computational budget. The loss of D2DMoE plateaus for higher budget levels, while the baseline displays consistently worse results whenever we lower the computational budget. Notably, for the larger Gemma-2B model our method performs well for most compute budgets, while the performance of MoEfication collapses (we analyze this in more detail in Appendix D).

5. Conclusion

We introduce Dense to Dynamic-k Mixture-of-Experts (D2DMoE), a novel approach that induces activation sparsity to improve the efficiency of Transformer-based models by converting their layers to Mixture-of-Experts (MoE). We demonstrate the interplay between the activation sparsity of dense models and the efficiency of converted MoEs. Moreover, we introduce regression-based router training and dynamic-k routing, which enable our method to efficiently utilize the induced sparsity. Our approach is compatible with the existing Transformer architectures and significantly improves upon existing MoE conversion schemes. Our findings contribute to the ongoing efforts to make Transformer models more efficient and accessible for a wider range of applications, especially in resource-constrained environments.

References

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Xuanyao Chen, Zhijian Liu, Haotian Tang, Li Yi, Hang Zhao, and Song Han. Sparsevit: Revisiting activation sparsity for efficient high-resolution vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2061–2070, June 2023.

Aidan Clark, Diego De Las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoffmann, Bogdan Damoc, Blake Hechtman, Trevor Cai, Sebastian Borgeaud, et al. Unified scaling laws for routed language models. In *International Conference on Machine Learning*, pages 4057–4086. PMLR, 2022.

Gonçalo M Correia, Vlad Niculae, and André FT Martins. Adaptively sparse transformers. *arXiv preprint arXiv:1909.00015*, 2019.

Erik Daxberger, Floris Weers, Bowen Zhang, Tom Gunter, Ruoming Pang, Marcin Eichner, Michael Emmersberger, Yinfei Yang, Alexander Toshev, and Xianzhi Du. Mobile v-moes: Scaling down vision transformers via sparse mixture-of-experts. *arXiv preprint arXiv:2309.04354*, 2023.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

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

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

Georgios Georgiadis. Accelerating convolutional neural networks via activation map compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7085–7095, 2019.

Aaron Gokaslan and Vanya Cohen. Openwebtext corpus. `http://Skylion007.github.io/OpenWebTextCorpus`, 2019.

Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7436–7456, 2021.

Patrik O Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research*, 5(9), 2004.

Sebastian Jaszczur, Aakanksha Chowdhery, Afroz Mohiuddin, Lukasz Kaiser, Wojciech Gajewski, Henryk Michalewski, and Jonni Kanerva. Sparse is enough in scaling transformers. *Advances in Neural Information Processing Systems*, 34:9895–9907, 2021.

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

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *ACM computing surveys (CSUR)*, 54(10s):1–41, 2022.

Mark Kurtz, Justin Kopinsky, Rati Gelashvili, Alexander Matveev, John Carr, Michael Goin, William Leiserson, Sage Moore, Nir Shavit, and Dan Alistarh. Inducing and exploiting activation sparsity for fast inference on deep neural networks. In *International Conference on Machine Learning*, pages 5533–5543. PMLR, 2020.

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

Zonglin Li, Chong You, Srinadh Bhojanapalli, Daliang Li, Ankit Singh Rawat, Sashank J Reddi, Ke Ye, Felix Chern, Felix Yu, Ruiqi Guo, et al. The lazy neuron phenomenon: On emergence of activation sparsity in transformers. In *The Eleventh International Conference on Learning Representations*, 2022.

Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, et al. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*, pages 22137–22176. PMLR, 2023.

Mikko I Malinen and Pasi Fränti. Balanced k-means for clustering. In *Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshop, S+ SSPR 2014, Joensuu, Finland, August 20-22, 2014. Proceedings*, pages 32–41. Springer, 2014.

Iman Mirzadeh, Keivan Alizadeh, Sachin Mehta, Carlo C Del Mundo, Oncel Tuzel, Golnoosh Samei, Mohammad Rastegari, and Mehrdad Farajtabar. Relu strikes back: Exploiting activation sparsity in large language models. *arXiv preprint arXiv:2310.04564*, 2023.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Zihan Qiu, Zeyu Huang, and Jie Fu. Unlocking emergent modularity in large language models. In *2024 Annual Conference of the North American Chapter of the ACL*, 2024.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34:8583–8595, 2021.

Stephen Roller, Sainbayar Sukhbaatar, Jason Weston, et al. Hash layers for large sparse models. *Advances in Neural Information Processing Systems*, 34:17555–17566, 2021.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2016.

Mingjie Sun, Xinlei Chen, J Zico Kolter, and Zhuang Liu. Massive activations in large language models. *arXiv preprint arXiv:2402.17762*, 2024.

Gemma Team. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

Hugo Touvron, Matthieu Cord, and Hervé Jégou. Deit iii: Revenge of the vit. In *European Conference on Computer Vision*, pages 516–533. Springer, 2022.

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

Shikhar Tuli and Niraj K Jha. Acceltran: A sparsity-aware accelerator for dynamic inference with transformers. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Bartosz Wójcik, Marcin Przewięźlikowski, Filip Szatkowski, Maciej Wołczyk, Klaudia Bałazy, Bartłomiej Krzepkowski, Igor Podolak, Jacek Tabor, Marek Śmieja, and Tomasz Trzciński. Zero time waste in pre-trained early exit neural networks. *Neural Networks*, 168:580–601, 2023.

Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. Moefication: Transformer feed-forward layers are mixtures of experts. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 877–890, 2022.

Zhengyan Zhang, Zhiyuan Zeng, Yankai Lin, Chaojun Xiao, Xiaozhi Wang, Xu Han, Zhiyuan Liu, Ruobing Xie, Maosong Sun, and Jie Zhou. Emergent modularity in pre-trained transformers. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4066–4083. Association for Computational Linguistics, July 2023.

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. *Advances in Neural Information Processing Systems*, 35: 7103–7114, 2022.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.

Simiao Zuo, Qingru Zhang, Chen Liang, Pengcheng He, Tuo Zhao, and Weizhu Chen. MoEBERT: from BERT to mixture-of-experts via importance-guided adaptation. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1610–1623, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.116. URL https://aclanthology.org/2022.naacl-main.116.

A. Limitations and Broader Impact

While D2DMoE displays promising results in reducing the computational cost of inference in Transformer models, a few limitations should be acknowledged. Our proposed sparsity enforcement and router training phases require additional training time. This overhead, while small, must be considered when evaluating the benefits of our approach. Moreover, we demonstrate improved performance over existing approaches on common NLP and CV tasks, but the scope of our experiments is restricted due to limited access to computational resources. Further research is needed to explore its applicability to extremely large models.

Our work focuses primarily on fundamental machine learning research and we do not see any specific risks or ethical issues associated with our method. Nevertheless, we recognize the potential for misuse of machine learning technology and advocate for responsible AI practices to mitigate such risks.

B. Related Work

Mixture-of-Experts. Sparse Mixture-of-Experts layers (MoE) consist of a predefined number of experts along with a gating network that routes each input to a subset of experts. MoE layers were introduced as an efficient way to further increase the capacity of deep neural networks applied for NLP tasks, initially in LSTM models (Shazeer et al., 2016), and later in Transformers (Lepikhin et al., 2020). Since then, they have also been applied to computer vision (Riquelme et al., 2021; Daxberger et al., 2023). MoE layers have gained significant popularity primarily due to their excellent scaling properties (Du et al., 2022; Clark et al., 2022). Nonetheless, training such models is challenging, primarily because gating decisions must be discrete to ensure sparse expert selection. Various methods of training were proposed, some of which include reinforcement learning (Bengio et al., 2013), weighting the expert output by the probability to allow computation of the gradient of the router (Shazeer et al., 2016), or using the Sinkhorn algorithm (Clark et al., 2022). Some of those approaches also suffer from the possibility of load imbalance and therefore require auxiliary losses or alternative expert selection methods (Fedus et al., 2022; Zhou et al., 2022). Interestingly, in many cases fixed routing functions perform similarly to trainable routers (Roller et al., 2021), which suggests that current solutions are largely suboptimal. MoE models can also be derived from pre-trained dense models by splitting the model weights into experts and independently training the routers for each layer (Zhang et al., 2022; Zuo et al., 2022), which avoids most of the problems present in end-to-end training.

Activation sparsity in Transformers. Li et al. (2022) show that ReLU-based Transformer models produce sparse activations in their intermediate representations, an effect that is prevalent across architectures, layers, and modalities. They propose a simple rule for keeping only top-k activations in each MLP layer, which results in a model with comparable performance. Similarly, Mirzadeh et al. (2023) demonstrate that ReLU activation function in LLMs encourages ensuing activation sparsity that can be leveraged to skip redundant computations. Tuli and Jha (2023) take a step further and design a dedicated Transformer architecture accelerator that also exploits activation sparsity, while Liu et al. (2023) proposes to predict activation sparsity structure in LLMs and reduce the model latency by skipping redundant computations. Jaszczur et al. (2021) demonstrate that it is possible to train Transformer models from scratch with a fixed level of activation sparsity and obtain similar performance. Finally, a related line of works focuses on sparsity in the attention distributions instead of intermediate representations (Correia et al., 2019). None of the abovementioned methods explore induced activation sparsity as a way to increase computational gains, nor do they address variance of the number of sparse activations on a per-token basis.

C. Difference between router training in D2DMoE and MoEfication

Our router training procedure is similar to the one proposed in MoEfication (Zhang et al., 2022), but the source code of the method provided by the authors[1] contains a different routing objective than the one reported in the paper. While the paper describes their router training objective as a prediction of the sum of ReLU activation values in each expert, the source code uses prediction labels created from the sum of the activations in the intermediate layer divided by the maximum value in the entire batch and minimize the binary cross-entropy loss. Assuming that $a_{k,j}$ is the activation vector in the hidden layer of expert j for sample k, the label generation for their router can be expressed as:

$$y_{k,j} = \frac{\sum_i a_{k,j,i}}{\max_{l,m} \sum_i a_{l,m,i}} \tag{5}$$

In comparison to their approach, the router training procedure in D2DMoE differs in multiple aspects:

- Our router considers the output of each expert instead of looking at the activations in the intermediate layers.

[1]MoEfication source code for router training is publicly available at: https://github.com/thunlp/MoEfication/blob/c50bb850307a36f8a0add6123f56ba309a156d13/moefication/utils.py#L188-L260

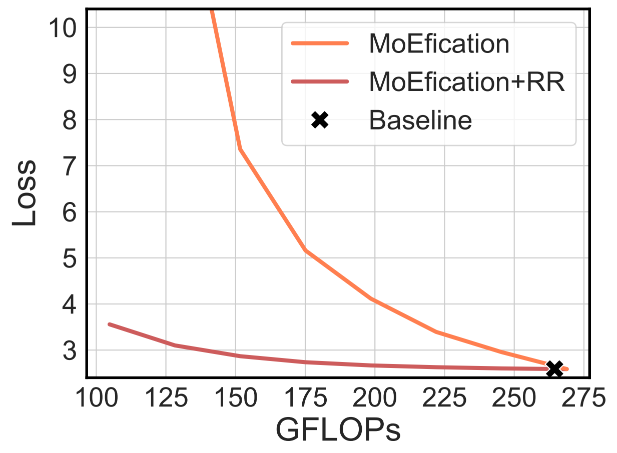


Figure 4: Comparision of performance on Gemma-2B for MoEfication with vanilla routing and with our regression routing.

- Instead of using artificially created labels based on the sums of activation values, we predict the ℓ^2-norm of the output. This has the additional benefit that our router can work with alternative activation functions.
- Our router is trained with the mean-squared error instead of the binary cross-entropy loss. The output of our router is constrained to positive values, while the MoEfication router is constrained to outputs in $[0, 1]$.

D. Routing analysis for large models

As presented in Figure 3c, in comparison to other considered benchmarks MoEfication visibly underperforms on language modeling with Gemma-2B. We attribute this to the emergence of massive activations in LLMs that reach a specific scale (Sun et al., 2024). Massive activations are outliers along certain feature dimensions whose magnitudes are thousands of times larger than the magnitudes of other activations. The training objective of MoEfication described in Equation (5) uses maximum activation over the entire batch to normalize the target label for each expert. Upon encountering large outlier values, those labels become effectively meaningless, as the values for most of the experts become very close to zero. In this case, the router effectively learns to output zero labels for most of the experts aside from the ones corresponding to the outlier values.

In comparison to MoEfication, our router training scheme does not make use of such normalization, and should therefore be robust to the emergence of massive activations. To validate this, we apply MoEfication on Gemma-2B, but with our regression routing instead of the original router training strategy. We compare the resulting model with vanilla MoEfication in Figure 4 and notice that replacing the routing scheme is enough for the model to learn effective expert assignment, as even though the expert choice is static and the base model is not sparsified, the cost-loss trade-off has significantly improved. This simple experiment shows that our regression routing objective is more robust than MoEfication when scaling to larger models.

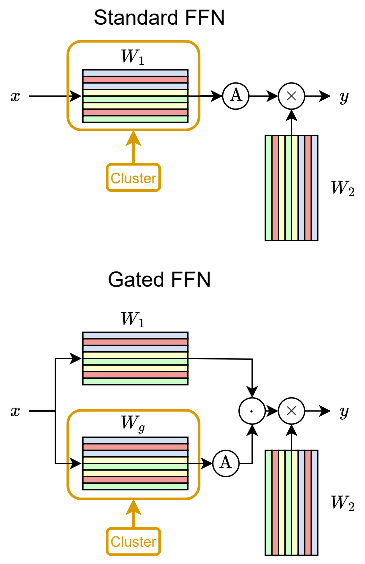


Figure 5: D2DMoE extension to Gated MLP.

E. D2DMoE extension to GLU-based layers

To provide better intuition behind the extension of our method to GLU-based gated MLPs mentioned in Section 3.2, we visualize the differences between standard FFN and Gated FFN and the application of our method in Figure 5. Standard Transformer MLP realizes the following function:

$$y(x) = \mathbf{W}_1 A(\mathbf{W}_2 x), \tag{6}$$

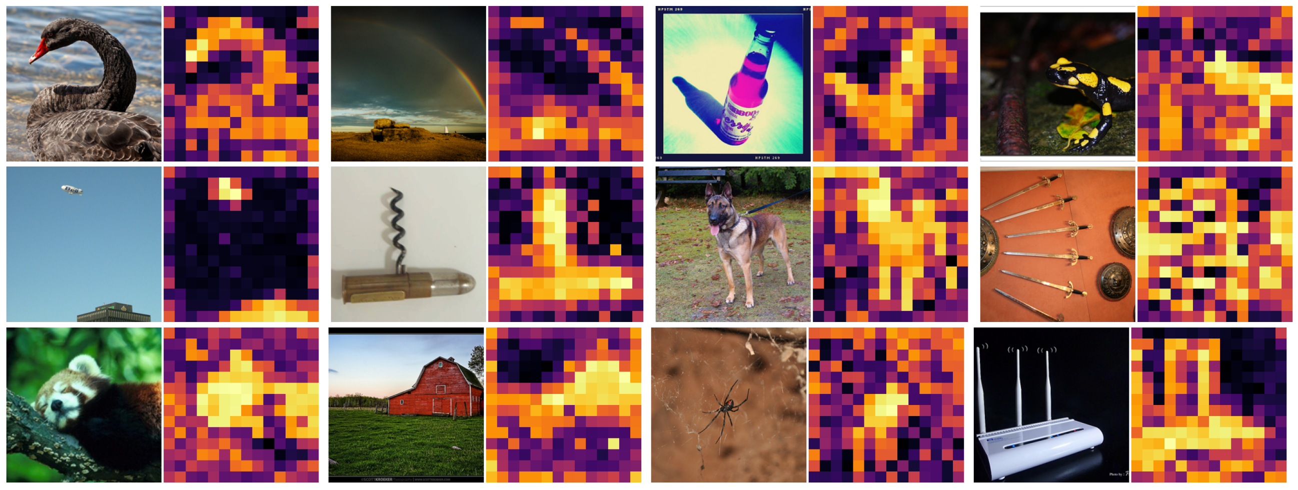
where $\mathbf{W}_1$, $\mathbf{W}_2$ are the weights for the upscale and downscale projections[2] and A stands for the activation function. In comparison, gated MLP can be written down as:

$$y(x) = \mathbf{W}_1 (A(\mathbf{W}_g x) \circ \mathbf{W}_2 x), \tag{7}$$

where $\mathbf{W}_g$ is the weight for the added gate projection.

The intuition behind MoEfication, which our method also follows for standard FFNs, is that the sparsity of the intermediate, post-activation representations determines the sparsity of the output representation. Therefore, the expert split is performed based on the weights of the upscale projection, as zeroed neurons in the upscale activations will also result in zeroed outputs of the downscale projection. When extending D2DMoE to Gated MLPs, our intuition

[2]We omit biases for simplicity.



(a) Computational load maps for ImageNet-1k sample images

Figure 6: Computational load maps of selected ImageNet-1k samples for our converted ViT-B model with $\tau = 0.0025$. D2DMoE allocates its computational budget to semantically important regions of the input.

is that the gating projections determine the sparsity of all the later representations, as both upscale and downscale are multiplied with the gating values. Therefore, we propose to build the experts through clustering performed on the gating weights $\mathbf{W}_g$ and use the indices obtained through expert split on gating weights to construct experts from $\mathbf{W}_1$ and $\mathbf{W}_2$. Following similar reasoning, for GLU-based models, we also perform activation sparsity enforcement on the gating projections instead of upscale projections as described originally in Section 3.1.

F. Expert selection patterns

D2DMoE allows the model to allocate different computational resources to various layers. We expect the model to allocate more compute to tokens containing information relevant to the current task. Since each token position in a ViT model corresponds to a separate and non-overlapping part of the input image, we can easily plot a heatmap to indicate the regions of the image where the model spends its computational budget. In Figure 6a we present such an analysis for our converted ViT-B model. As expected, the dynamic-k routing enables the model to minimize the computational effort spent on regions that contain insignificant information.

G. Training and hardware details

In this Section, we describe the technical details used in the D2DMoE conversion procedure. For full reproducibility, we share the source code that we used for conducting the experiments. All experiments were performed using the `PyTorch` library (Paszke et al., 2019) on the NVIDIA A100 and V100 GPUs on internal clusters. We utilize the *fvcore* library to count model FLOPs[3].

G.1. Image classification

All methods start with the same pre-trained ViT-B from the `torchvision` [4] library and are trained on ImageNet-1k using the augmentation proposed by Touvron et al. (2022). We use mixup (0.8), cutmix, label smoothing (0.1), gradient clipping (1.0) and the Adam optimizer with a cosine learning rate schedule without warm-up. We finetune the model for 90 epochs with sparsity enforcement weight $\alpha = 0.2$, initial learning rate $2 \cdot 10^{-5}$ and batch size 512. We then convert the modules into MoE layers, and train the gating networks for 7 epochs with the initial learning rate set to 0.001 and batch size 128. We train ZTW for 100 epochs in total, allocating 5 epochs for ensemble training, while keeping the rest of the original hyperparameters unchanged. For MoEfication, we first convert the pre-trained model to ReLU-based one and finetune for 90 epochs with an initial learning rate of 0.0001 and batch size 256. We then split the weights and train the routers for 10 epochs with the initial learning rate 0.001 and batch size 256.

G.2. Language modeling

We base our code and hyperparameters for GPT2-base on the *nanoGPT* repository provided at `https://github.com/karpathy/nanoGPT`. We initial-

[3] `https://github.com/facebookresearch/fvcore`

[4] `https://pytorch.org/vision/stable/models.html`

ize the model from https://huggingface.co/openai-community/gpt2. In all pretraining experiments, we initialize models from a publicly available OpenAI checkpoint pre-trained on a closed-source WebText dataset and finetune for the fixed number of 1000 steps with the effective batch size equal to the value in the repository through gradient accumulation. We train the routers for D2DMoE and MoEfication for 2000 steps using one GPU and tuning the learning rates for a given expert size from the range between $0.002 - 0.005$. For router training, we use Adam optimizer and cosine warmup scheduler.

For Gemma-2B, we start from the checkpoint at https://huggingface.co/google/gemma-2b. We also finetune the model for 1k steps with an effective batch size of 1024, sequence length of 1024 and Adam optimizer with a learning rate of 1e-4. As Gemma's hidden dimension is much larger than the other considered models, we change the hidden dimensionality of the routers to 512 for both our method and MoEfication, but keep the other hyperparameters the same as in the rest of the experiments. For MoEfication, Gemma, we use 512 experts to obtain an expert size comparable to the one in their paper. For our method, we use 2048 experts. In D2DMoE, we set sparsity enforcement weight to 0.00003. We train the routers for 500 steps with Adam and effective batch size of 16 and use a learning rate of 0.001.