# Exploiting Transformer Activation Sparsity with Dynamic Inference

**Mikołaj Piórczyński**
Warsaw University of Technology
Artificial Intelligence Society Golem

**Filip Szatkowski**
Warsaw University of Technology
IDEAS NCBR

**Klaudia Bałazy**
Jagiellonian University

**Bartosz Wójcik**
Jagiellonian University
IDEAS NCBR

## Abstract

Transformer models, despite their impressive performance, often face practical limitations due to their high computational requirements. At the same time, previous studies have revealed significant activation sparsity in these models, indicating the presence of redundant computations. In this paper, we propose Dynamic Sparsified Transformer Inference (DSTI), a method that radically reduces the inference cost of Transformer models by enforcing activation sparsity and subsequently transforming a dense model into its sparse Mixture of Experts (MoE) version. We demonstrate that it is possible to train small gating networks that successfully predict the relative contribution of each expert during inference. Furthermore, we introduce a mechanism that dynamically determines the number of executed experts individually for each token. DSTI can be applied to any Transformer-based architecture and has negligible impact on the accuracy. For the BERT-base classification model, we reduce inference cost by almost 60%.

## 1 Introduction

In recent years, Transformer [1] became a go-to model architecture in many fields of deep learning such as natural language processing [2, 3] or computer vision [4, 5]. Those models often have a large number of parameters [3, 6], which gives them sufficient expressivity and enables them to effectively accumulate knowledge. However, despite their impressive performance [3, 7], they require costly high-end computational resources, and their applications are limited due to high latency and energy consumption. Simultaneously, sparse Mixture-of-Experts (MoE) models have gained significant attention as a compelling approach for enhancing model expressiveness and capacity [8]. Unlike their dense counterparts, these models are able to handle a much larger number of parameters with only a slight increase in processing time, and many of the latest state-of-the-art Transformer models use MoE layers [9, 6, 10]. Unfortunately, training MoE models from scratch may be unstable and is prone to expert imbalance or representation collapse [8, 11], which limits their applicability.

In this paper, we follow the recently introduced approach of turning dense models into sparse MoE models [12] and propose Dynamic Sparsified Transformer Inference (DSTI), a simple and practical way to significantly reduce the computational cost of the inference in Transformer models. Inspired by the recent works that show the benefits of the natural sparsity emerging in the Transformer models [13], we propose to train the dense models with an additional loss component that enforces activation sparsity. Then, we construct the MoE layers by splitting the dense matrices of FFN layers into experts and subsequently training small gating networks. Moreover, we propose a novel learning objective for training the routers that enables them to accurately predict the relative contribution of

Preprint. Under review.

each expert. Finally, we introduce Dynamic-$k$ routing, which allows the model to adapt the amount of compute to the difficulty of the input, which increases its efficiency even further. DSTI achieves performance close to the original dense model while using only a fraction of its computational resources.

## 2 Related Works

**Mixture-of-Experts models** Sparse MoE was first proposed for RNNs by Shazeer et al. [8] and since then has been successfully applied in the NLP domain [14, 6]. Recently, those models have also been gaining popularity in computer vision [9, 15]. Sparse Mixture-of-Experts (MoE) transformers replace the FFN layers with multiple experts, which themselves are smaller feedforward networks, and a router that selects which experts to use for the current input. This change significantly increases model capacity, while inducing only a small computational overhead through the use of the router. Additionally, recent research suggests that MoE models have favorable properties in the context of the scaling laws [16].

**Sparsification of Dense Transformers** Several works notice the difficulties of end-to-end training of MoE models and propose alternative methods to obtain Sparse MoE more efficiently. Methods such as EvoMoE [17] or Sparse Upcycling [18] propose to progressively make the model sparser over the course of the training. Other works observe that activation patterns in Transformers are highly sparse [12, 13], and propose to take advantage of this phenomenon without training. Notably, Zhang et al. [12] introduced *MoEfication*, a method that enhances the computational efficiency of Transformer models. Our method builds on MoEfication and follows the expert construction scheme proposed in this paper.

## 3 Method

DSTI is a three-step method for obtaining an efficient Transformer MoE model. The first stage of our method consists of fine-tuning a pre-trained model with an auxiliary loss that enforces activation sparsity. The FFN modules in every layer are then divided into experts, and at the last step we train the routing networks that enable dynamic selection of experts. In this section, we describe all the components of DSTI in detail.

**Enforcing activation sparsity** The scheme of reducing inference cost by dividing the model into independently activated modules relies on the well-known phenomenon of activation sparsity exhibited by most deep neural networks [19], especially Transformer architecture-based models [13]. Taking inspiration from this observation, we anticipate that enforcing activation sparsity with an auxiliary loss may allow for execution of an even smaller number of experts, resulting in overall computational savings. As such, we propose to apply the $\ell_1$ norm penalty on the feature representations in the middle layer of each FFN module:

$$L_s(x) = \frac{1}{L} \sum_{l=1}^{L} ||a^l||_1, \tag{1}$$

where $a^l$ is the activation tensor from the middle layer of the $l$-th FFN for input $x$, and $L$ is the number of Transformer blocks. Overall, the model is trained with the following cost function:

$$L(x) = L_{\text{CE}}(\hat{y}, y) + \alpha_s L_s(x) \tag{2}$$

where $L_{CE}$ is the standard cross-entropy loss, and $\alpha_s$ is the hyperparameter for scaling the sparsity enforcement loss. While this loss could be applied during pretraining, in practice we add it during finetuning of the model so that application to pretrained models is still possible.

**Expert construction** We construct the experts using parameter clustering method proposed by Zhang et al. [12], which we briefly describe here for the convenience of the reader. Weights of each neuron from the first matrix $W_1$ are treated as its features and are fed into a balanced $k$-means algorithm [20]. The resulting cluster indices are used to split the first linear layer $W_1$, the first bias vector $b_1$, and the second linear layer $W_2$ into $E$ experts. The second bias $b_2$ is not affected by this procedure. The process is repeated for each FFN block.

**Regression routing objective** 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, similarly to Zhang et al. [12]. However, instead of framing the problem as a classification task[1], our gating network directly predicts the sum of activations in the hidden layer of each $i$-th expert $s_i = \sum_j a_{ij}$. We train the gating network using the standard mean squared error. Assuming ReLU activation function is used, $s_i$ is always positive, and to ensure the positive output of the gating network, we take the absolute value of the gating network output. The regression-based formulation is still compatible with commonly used top-$k$ expert selection, but enables more precise attribution of the contribution of each expert, as we show later in the experiments section.

**Dynamic-$k$ gating** Commonly used MoE layers always execute $k$ top 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 [21] or token [8]. However, cognitive studies show that humans treat the data samples differently depending on their complexity and spend significantly less time on the easy samples [22]. Similarly, various conditional computation methods adjust their computational load to the difficulty of the input sample[23, 24]. Inspired by this, we modify the expert selection mechanism to allow for a varying number of experts. Since our gating network $g$ approximates the actual contribution of every expert $s \approx \hat{s} = g(x)$, we use those predictions to determine $k$. For each token, we set:

$$k = \min\{n \in \{1, ..., E\} | \sum_{i=1}^{n} \text{sort}(h)_i > \tau\}, \quad h_i = \frac{\hat{s}_i}{\sum_{j=1}^{E} \hat{s}_j} \tag{3}$$

where $\tau \in (0.0, 1.0)$ is a threshold that determines the preferred performance vs. computational cost trade-off. Note that after model deployment, $\tau$ can be adjusted anytime without the need for retraining.

# 4 Experiments

We evaluate the proposed method on *emotion* classification dataset [25] using BERT-base model [2] with ReLU activation function, and compare it with *MoEfication* [12]. All of the models finetuned in our experiments start from the same pretrained checkpoint. We use the parameter clustering and MLP router variant of MoEfication. We set the number of experts to 128 and use 2-layer MLP routers with a hidden size of 128. See the supplementary material for the full list of training hyperparameters.

To demonstrate the contribution of each piece of DSTI, we train additional variants of our method with different components ablated out and show the results of our study averaged over three random seeds in Figure 1. The models are evaluated in terms of task performance at different compute budgets, adjusting $k$ for methods with static Top-$k$ expert selection, or $\tau$ in case of Dynamic-$k$. For the comparison, we also provide the score of the standard dense BERT-base model. It can be seen that the proposed DSTI offers a significantly better trade-off between computational cost and accuracy than MoEfication, and each of its components plays a substantial role in the final performance. We emphasize that due to the widespread availability of efficient MoE layer implementations, the presented results translate to real speedups on both CPUs and GPUs.

## 4.1 Expert activation patterns

To explore the scale of variability of the computational effort introduced by Dynamic-$k$ routing, we investigate the distribution of executed expert counts in different layers of the model. Figure 2 shows the selection frequency of a given fraction of experts for various $\tau$ thresholds for DSTI trained with and without sparsity enforcement. As expected, models with higher activation sparsity require a smaller number of experts to meet the defined threshold. It is important to highlight the range of executed experts count, which for exemplary $\tau = 0.75$ with sparsity enforcement can vary between $5\%$ and $40\%$ in most of the layers. This suggests that computational adaptability mechanisms are crucial for efficient inference in Transformer-based models.

[1] It is worth noting that Zhang et al. [12] states that their learning objective is prediction of the sum of positive values in each expert, but in the source code for the paper they instead frame it as a classification problem and use modified binary cross entropy loss to train routers.

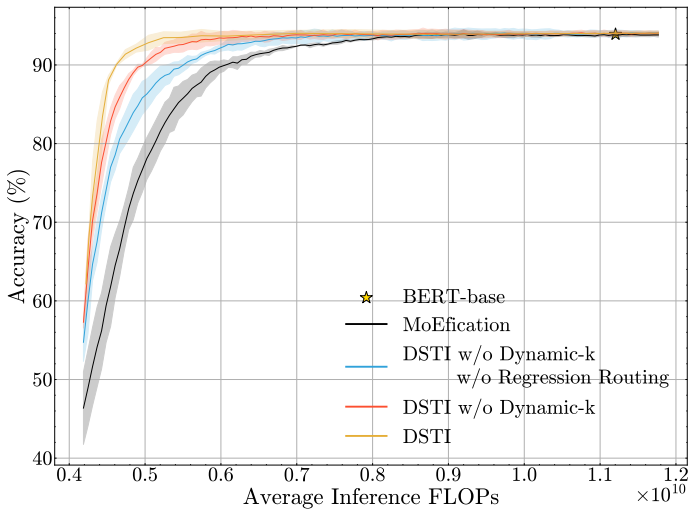


Figure 1: Accuracy vs. averaged computational cost for BERT-base, MoEfication, DSTI, and the ablated variants on *emotion* task. DSTI demonstrates superior performance on every considered computational budget, and each of its components improves the performance upon the MoEfication baseline.

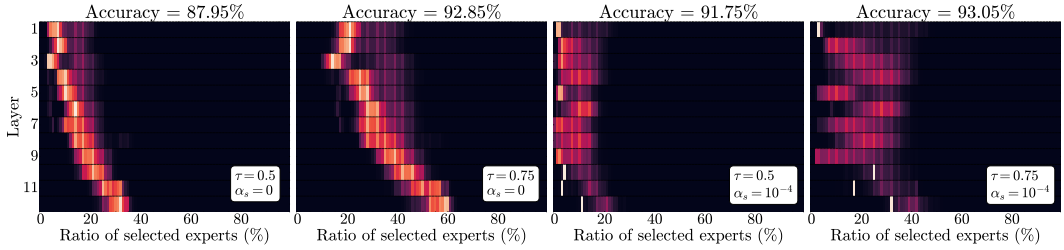


Figure 2: Distribution of the number of executed experts in each layer. The high variability of that number explains the computational gains from using Dynamic-$k$.

## 5 Conclusions

In this paper, we have proposed DSTI, a method that obtains computationally effective Transformer MoE models through enforcing activation sparsity, training routers with a novel regression objective, and using Dynamic-$k$ gating. Our approach demonstrates that activation sparsity is a key factor for achieving efficient inference. With Dynamic-$k$ gating we show that the intuition with different inputs having varying levels of difficulty is also true in deep learning models, and it is wasteful to assume a fixed amount of computation for each input. DSTI outperforms a simpler sparsification method, MoEfication, on various compute budgets and reduces the cost of inference by almost 60% with negligible impact on model accuracy.

### 5.1 Limitations and Future work

Following the previous works, we conduct our experiments using a ReLU-based model. While Zhang et al. [12] showed that a GELU-based model could be converted to a ReLU-based one, we would like to adopt DSTI to work with any Transformer-based model without conversion. Losses that enforce activation sparsity could be a promising direction to achieve this goal. Moreover, we would like to extend the analysis of our method to different tasks and modalities beyond text classification to show its generality. Finally, as we believe our method is orthogonal to the other inference speed-up methods, such as quantization or early-exits, we would like to explore the interplay between DSTI and those methods.

## Acknowledgments

We gratefully acknowledge Poland's high-performance Infrastructure PLGrid (HPC Centers: ACK Cyfronet AGH, PCSS, CI TASK, WCSS) for providing computer facilities and support within computational grant no. PLG/2023/01632. This research was also carried out with the support of the Laboratory of Bioinformatics and Computational Genomics and the High-Performance Computing Center of the Faculty of Mathematics and Information Science at Warsaw University of Technology. Filip Szatkowski is supported by the National Centre of Science (NCP, Poland) Grant No. 2022/45/B/ST6/02817. The work of Klaudia Bałazy was supported by the National Centre of Science (Poland) Grant No. 2020/39/D/ST6/01332. Klaudia Bałazy is affiliated with Doctoral School of Exact and Natural Sciences at the Jagiellonian University.

## 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. 2017. URL `https://arxiv.org/pdf/1706.03762.pdf`.

[2] 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.

[3] 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. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[4] 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. *arXiv preprint arXiv:2010.11929*, 2020.

[5] 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.

[6] 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.

[7] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[8] 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.

[9] 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.

[10] 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.

[11] Zewen Chi, Li Dong, Shaohan Huang, Damai Dai, Shuming Ma, Barun Patra, Saksham Singhal, Payal Bajaj, Xia Song, Xian-Ling Mao, et al. On the representation collapse of sparse mixture of experts. *Advances in Neural Information Processing Systems*, 35:34600–34613, 2022.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] Xiaonan Nie, Xupeng Miao, Shijie Cao, Lingxiao Ma, Qibin Liu, Jilong Xue, Youshan Miao, Yi Liu, Zhi Yang, and Bin Cui. Evomoe: An evolutional mixture-of-experts training framework via dense-to-sparse gate. *arXiv preprint arXiv:2112.14397*, 2021.

[18] Aran Komatsuzaki, Joan Puigcerver, James Lee-Thorp, Carlos Riquelme Ruiz, Basil Mustafa, Joshua Ainslie, Yi Tay, Mostafa Dehghani, and Neil Houlsby. Sparse upcycling: Training mixture-of-experts from dense checkpoints. In *The Eleventh International Conference on Learning Representations*, 2022.

[19] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings, 2011.

[20] 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.

[21] 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.

[22] Radu Tudor Ionescu, Bogdan Alexe, Marius Leordeanu, Marius Popescu, Dim P Papadopoulos, and Vittorio Ferrari. How hard can it be? estimating the difficulty of visual search in an image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2157–2166, 2016.

[23] Kaiyuan Liao, Yi Zhang, Xuancheng Ren, Qi Su, Xu Sun, and Bin He. A global past-future early exit method for accelerating inference of pre-trained language models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2013–2023, 2021.

[24] Thomas Verelst and Tinne Tuytelaars. Dynamic convolutions: Exploiting spatial sparsity for faster inference. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pages 2320–2329, 2020.

[25] Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. CARER: Contextualized affect representations for emotion recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3687–3697, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1404. URL `https://www.aclweb.org/anthology/D18-1404`.

[26] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[27] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

# Appendix

### Training details

In all our experiments we finetune model for 10 epochs using AdamW [26] algorithm with a $2 \times 10^{-5}$ learning rate and batch size 64. Then, we select the best model according to the accuracy on the held-off validation dataset. For training routers we set constant learning rate $10^{-3}$ and train them for 20 epochs with Adam optimizer [27] and batch size $512$. We set $\alpha_s = 10^{-4}$, which significantly increases model sparsity on average from $13\%$ of non-zero activations in the FFN layer to $1\%$, while preserving model performance.

### Enforcing activation sparsity

Table 1: Enforcing activation sparsity in BERT-base with different regularization strength $\alpha_s$. We can see that applying minor regularization significantly increases model sparsity without diminishing the performance. These results are consistent with prior works [13].

| $\alpha_s$ | Accuracy | Sparsity |
|---|---|---|
| — | 93.90% | 13.05% |
| $10^{-6}$ | 94.80% | 4.29% |
| $10^{-5}$ | 94.05% | 1.95% |
| $10^{-4}$ | 93.75% | 0.90% |