Parameter Efficient Finetuning for Reducing Activation Density in Transformers

Bharat Runwal[1] **Tejaswini Pedapati**[2] **Pin-Yu Chen**[2]

[1]**Independent Researcher,** [2] **IBM Research**

Abstract

Pretrained Language Models (PLMs) have become the de facto starting point for fine-tuning on downstream tasks. However, as model sizes continue to increase, traditional fine-tuning of all parameters becomes challenging. To address this, parameter-efficient fine-tuning (PEFT) methods have gained popularity as a means to adapt PLMs effectively. In parallel, recent studies have revealed the presence of activation sparsity within the intermediate outputs of the MLP blocks in transformers. Low activation density enables efficient model inference on sparsity-aware hardware. Building upon this insight, in this work, we propose a novel density loss that encourages higher activation sparsity (equivalently, lower activation density) in the pre-trained models. In our experiments, we demonstrate the effectiveness of our proposed approach **DEFT** by employing mainstream PEFT techniques like LoRA, Adapter, Prompt/Prefix Tuning. DEFT consistently achieves substantial reductions in activation density. For example, on the T5-Base model, DEFT leads to reductions of average **47.77%** in encoder density and **81.82%** in decoder density compared to PEFT. These trends are mirrored across various GeLU activation-based models, including ViT-Base (86M), ViT-Large (307M), RoBERTa-Base (125M), RoBERTa-Large (355M), and GPT2 (117M), with density reductions ranging from **29.61%** to **56.68%**.

1 Introduction

With the advent of pre-trained Large language models [1, 2, 3], fine-tuning [4] these models to adapt to a task has become prevalent. These models, with billions of parameters, demand substantial time, energy, and memory, leading to a significant environmental impact [5]. To mitigate this, parameter-efficient fine-tuning techniques have emerged [6, 7, 8, 9]. While various approaches focus on pruning [10, 11], distillation [12], or quantization [13, 14]. In contrast to all these techniques, we accelerate the model inference by increasing the activation sparsity. This is achieved by modifying the loss function to penalize high activation density.

Recent studies [15, 16] have uncovered a notable trait within the transformer architecture. Specifically, in the intermediate outputs of MLP (Multi-Layer Perceptron) blocks with ReLU activations, only a fraction of neurons activate for a given input, introducing sparsity in activation maps. Building on this, we propose a density loss to enhance activation sparsity in pre-trained models, effectively reducing activation density.

This induced sparsity bears significant energy-saving potential, especially on specialized hardware like ASICs (Application Specific Integrated Circuits) [17], which capitalize on zero-skip operations. By promoting sparsity, unnecessary computations on zero-valued activations are skipped, leading to reduced power consumption and more efficient model inference. This approach is particularly advantageous in resource-constrained environments or applications with strict energy constraints.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

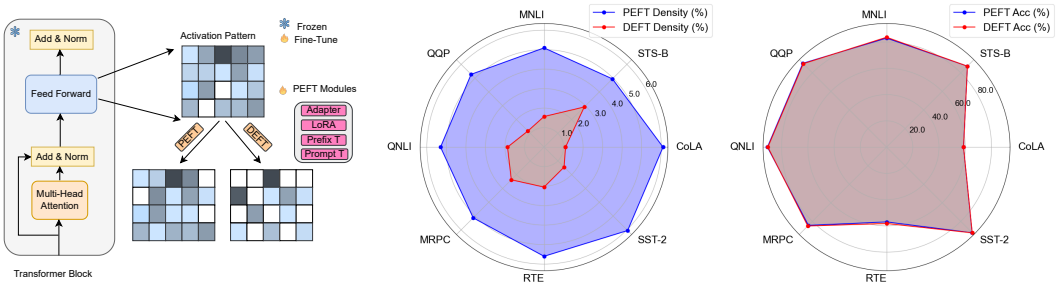

Figure 1: (a) Comparison between the activation patterns (in the intermediate output of MLP) after adapting to downstream tasks with PEFT and our proposed DEFT method. (b, c) Comparison of Activation Density (%) and Accuracy (%) for GLUE benchmark datasets using $\text{BERT}_{\text{BASE}}$ with Adapter for PEFT and DEFT.

In this work, we delve into the concept of activation sparsity and its potential benefits in optimizing transformer models. We present Density Efficient Fine-Tuning (DEFT) and demonstrate its effectiveness through experiments on well-known benchmarks, showcasing reduced activation density while maintaining competitive performance across diverse downstream tasks. We provide the illustration of our proposed method DEFT in 1a and we showcase the strength of our proposed method DEFT compared to PEFT in Figure (1b, 1c) for 8 datasets from GLUE benchmark when adapting $\text{BERT}_{\text{BASE}}$ model with the adapter module. Our proposed method DEFT leads to sparser activation patterns compared to PEFT. Figure 1b displays the average density (%) in MLP layers, while 1c shows the accuracy (%) on the validation dataset. DEFT maintains competitive performance with PEFT while reducing activation density.

To the best of our knowledge, we are the first to demonstrate that a significant degree of activation sparsity can be attained using a small number of trainable parameters. This is particularly notable in GeLU models. Prior studies primarily concentrated on ReLU-based models for investigating activation sparsity [17], as ReLU tends to induce more sparsity in activation maps. Our approach, combining parameter-efficient fine-tuning (PEFT) and activation sparsity, paves the way for resource-friendly transformer models across various applications.

2 Methodology

2.1 Background and Notations

In Transformers, the position-wise feed-forward networks employ a two-layer MLP. We measure the activation sparsity at the intermediate output of this two-layer MLP, following the works of [16] and [15]. Consider an input $X \in \mathbb{R}^{B \times K \times d_{\text{model}}}$, where B is the batch size, K is the sequence length and d_{model} denotes the dimensionality of the input features. Given an input matrix X, the output of the two-layer MLP can be described as:

$$Y(X; W^{(1)}, W^{(2)}) = f\left(XW^{(1)}\right) W^{(2)} \tag{1}$$

Here $W^{(1)} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{ff}}}$ and $W^{(2)} \in \mathbb{R}^{d_{\text{ff}} \times d_{\text{model}}}$ are the learnable parameters of the MLP layers. d_{ff} represents the hidden dimension of the MLP block, and f is the non-linear activation function.

To measure the sparsity of neuron activations, we first define the activation pattern as :

$$O = f\left(XW^{(1)}\right) \tag{2}$$

The matrix $O \in \mathbb{R}^{B \times K \times d_{\text{ff}}}$ is the activation pattern. Following [16], we can define the vector $s \in \mathbb{R}^{d_{\text{ff}}}$ as the average across the batches and sequence length of matrix O to represents the final feature map. So, we can measure the sparsity of neurons by counting the number of non-zeros in the feature map s.

2.2 DEFT: Parameter and Activation Density Efficient Fine-tuning

In this section, we introduce our proposed Density loss in DEFT. Our goal is to reduce the activation density (or increase activation sparsity) in MLP blocks for given inputs.

Previous work by [16] used a step function to count the number of positive elements precisely, but this operation is non-differentiable and cannot be used for our purpose of reducing activation density in an end-to-end learning setup. Therefore, to approximate the number of non-zero entries in the sparse vector s, we use the hyperbolic tangent function with a scaling parameter β for ReLU-activation based models, [18] used the similar function for their purpose of generating adversarial inputs for sparsity attacks. We also use a differentiable approximation of the l_0 norm [17] for GeLU and other activation based models.

$$\tanh(x, \beta) = \frac{e^{\beta \cdot x} - e^{-\beta \cdot x}}{e^{\beta \cdot x} + e^{-\beta \cdot x}} \tag{3}$$

$$\hat{l}_0(x) = \sum_{i=1}^{n} \left(\frac{x_i^2}{x_i^2 + \epsilon} \right) \tag{4}$$

By adjusting the value of β in Equation 3, we can control the abruptness to approximate the step function (and therefore sparsity) of the values. Higher values of β make the function more closely resemble the step function. In Equation 4, $\epsilon \in \mathbb{R}$ is a parameter dictating the quality of the approximation—lower values of ϵ correspond to better approximations.

We can define the density loss $\mathcal{L}_{\text{density}}(x)$ with tanh approximation as follows:

$$\mathcal{L}_{\text{density}}(x) = \frac{1}{n} \sum_{L} \sum_{i} \tanh(s_{l_i}, \beta) \tag{5}$$

Here, n is the total number of neurons in all the MLP layers, L is the total number of layers in the transformer and s_l is the final feature map as defined earlier for the layer l. The summation is across all the layers and the elements of the vector s_l. We can similarly define the loss with $\hat{l_0}$ norm.

In DEFT, we fine-tune the model for downstream tasks by exclusively training specialized parameter-efficient modules, freezing the pretrained transformer parameters. DEFT leverages established PEFT techniques like Prompt Tuning, Prefix Tuning, Adapters, and LoRA (details in Supplementary material).

For DEFT, we solve the following optimization problem:

$$\arg\min_{\Phi} \mathcal{L}_{\text{total}} = \mathcal{L}_{\text{T}}\left(\text{D}; \{\Theta, \Phi\}\right) + \alpha \cdot \mathcal{L}_{\text{density}}\left(\text{D}; \{\Theta, \Phi\}\right)$$

Here, Φ are tunable parameters, and Θ denotes the frozen pre-trained parameters. $\mathcal{L}_{\text{T}}$ integrates task-specific objectives on dataset D. α balances task performance and activation sparsity induction. Higher α values encourage sparser activations. Refer to Supplementary material Algorithm 1 for the DEFT algorithm.

The tunable parameters Φ encompass a range of modules like Adapters, LoRA, Prefix-Tuning, and Prompt-Tuning. By introducing a small fraction of trainable parameters (a few % only), we trigger activation sparsity within the MLP blocks, yielding twofold efficiency gains: (1) Activation Sparsity for hardware acceleration, and (2) Efficient training and storage, reducing both time and memory usage without compromising task performance.

3 Experiments

Our method was rigorously tested on a range of datasets: GLUE [19] for BERT; For {RoBERTa, GPT2} we used SST-2 dataset from GLUE; SQuAD [20] for question answering (T5), and CIFAR-10 [21] for vision tasks (ViT). We employed a suite of techniques including Adapter, LoRA, Prefix-Tuning, and Prompt Tuning for BERT; and {Adapter, LoRA} for T5 models. For GeLU-based models, we used an Adapter for our experiments. Further details about datasets, models and evaluation metrics can be found in the supplementary material. The results were averaged across 5 random seeds.

In addition to task-specific metrics, we evaluated activation sparsity. This was quantified by computing the density (%) of non-zero values in the intermediate activation matrix of the MLP block for each layer. The average density across all layers and the validation set was reported. We also report the Density Change (%) across different datasets and methods, similar to the energy consumption ratio in [22], which is calculated as:

$$\text{Density Change}(\%) = \left(\frac{\text{Density}_{\text{PEFT}} - \text{Density}_{\text{DEFT}}}{\text{Density}_{\text{PEFT}}} \right) \times 100$$

Method	Performance	ViT-Base (86M)	ViT-Large (307M)	RoBERTa-Base (125M)	RoBERTa-Large (355M)	GPT2 (117M)
	Trainable (%)	(2.04%)	(2.04%)	(1.17%)	(1.17%)	(1.87%)
PEFT	Metric (↑)	98.13 ± 0.03	99.04 ± 0.06	94.19 ± 0.24	96.02 ± 0.54	88.38 ± 0.19
	Density (↓)	84.84 ± 0.09	99.91 ± 0.00	99.59 ± 0.0	93.90 ± 0.06	99.96 ± 0.00
DEFT	Metric (↑)	97.75 ± 0.09	98.31 ± 0.08	94.15 ± 0.34	95.80 ± 0.39	88.34 ± 0.39
	Density (↓)	77.74 ± 0.09	70.33 ± 2.47	70.78 ± 0.04	40.68 ± 0.29	70.51 ± 1.15
	Density Change (%) (↑)	8.38	29.61	28.93	56.68	29.46

Table 1: Peformance Comparison on different GeLU models using Adapter module with PEFT and DEFT.

Model	Module (% Trainable)	Loss Type	SQuAD			
			F1	Exact-Match	Enc-Density	Dec-Density
T5- Small (60M)	Adapter (0.33%)	PEFT	82.58 ± 0.08	74.48 ± 0.07	4.76 ± 0.01	4.07 ± 0.03
		DEFT	82.41 ± 0.11	74.19 ± 0.13	3.51 ± 0.07	1.95 ± 0.01
		Density Change(%)			26.26	52.08
	LoRA (0.96%)	PEFT	82.60 ± 0.06	74.54 ± 0.10	4.80 ± 0.01	3.97 ± 0.01
		DEFT	82.38 ± 0.09	74.19 ± 0.13	3.33 ± 0.02	1.51 ± 0.02
		Density Change(%)			30.62	61.96
T5- Base (220M)	Adapter (0.40%)	PEFT	88.28 ± 0.04	81.19 ± 0.05	2.64 ± 0.02	3.22 ± 0.04
		DEFT	88.21 ± 0.04	81.08 ± 0.12	1.61 ± 0.03	0.96 ± 0.05
		Density Change(%)			39.01	70.19
	LoRA (0.78%)	PEFT	88.33 ± 0.03	81.30 ± 0.02	2.70 ± 0.01	3.19 ± 0.01
		DEFT	88.42 ± 0.04	81.40 ± 0.05	1.41 ± 0.01	0.58 ± 0.002
		Density Change(%)			47.77	81.82

Table 2: Performance comparison of different methods on Question Answering Dataset (SQuAD) with T5_{SMALL} and T5_{BASE} .

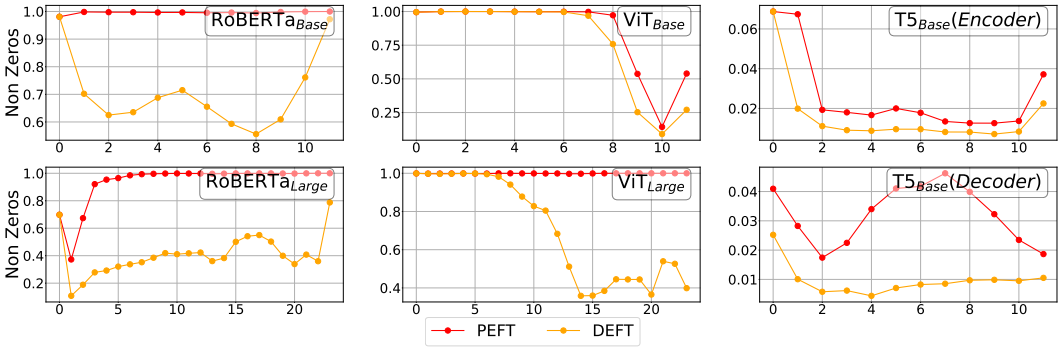

Figure 2: **Layerwise Non-zeros (density):** Layerwise non-zeros for different models with Adapter module on the validation set (For CIFAR-10, we used Test-set) of different tasks following Table 1.

3.1 Results

The performance using the Adapter module and $\tanh$ approximation on the GLUE benchmark with the $\text{BERT}_{\text{BASE}}$ (ReLU) model is summarized in Fig. 1 (b,c) and detailed in Table 3 of supplementary material with extra results on the different modules. For DEFT, activation sparsity is enhanced with marginal to no impact on downstream performance. Tables 1 and 2 present the performance of various GeLU and T5 models respectively. Notably, DEFT consistently reduces activation density across all models, showcasing its effectiveness. Larger models, in particular, demonstrate more pronounced sparsity patterns with DEFT, for example, with RoBERTa-Base we achieve a Density-Change(%) of 28.93 while RoBERTa-Large achieves 56.68 with the same number of trainable parameters.

In Figure 2, we provide detailed layerwise density plots. These visualizations illustrate DEFT's success in promoting activation sparsity across layers in different models.

4 Conclusion

In this work, we introduced DEFT, an innovative extension to PEFT designed to induce activation sparsity in MLP layers of frozen pre-trained transformer blocks. Through extensive experiments on diverse datasets spanning language and vision modalities, we demonstrated DEFT's capacity to significantly reduce activation density without compromising downstream performance compared to PEFT. This contribution opens up fresh avenues for density-efficient PEFT of pretrained language models.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805, 2019.

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

[3] 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. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020.

[4] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 328–339. Association for Computational Linguistics, 2018.

[5] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3645–3650. Association for Computational Linguistics, 2019.

[6] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[7] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 3045–3059. Association for Computational Linguistics, 2021.

[8] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4582–4597. Association for Computational Linguistics, 2021.

[9] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR, 2019.

[10] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Snip: single-shot network pruning based on connection sensitivity. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[11] Hidenori Tanaka, Daniel Kunin, Daniel L. K. Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[12] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019.

[13] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm.int8(): 8-bit matrix multiplication for transformers at scale. *CoRR*, abs/2208.07339, 2022.

[14] Ali Hadi Zadeh, Isak Edo, Omar Mohamed Awad, and Andreas Moshovos. GOBO: quantizing attention-based NLP models for low latency and energy efficient inference. In *53rd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2020, Athens, Greece, October 17-21, 2020*, pages 811–824. IEEE, 2020.

[15] Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. Moefication: Transformer feed-forward layers are mixtures of experts. In *Findings*, 2021.

[16] Zonglin Li, Chong You, Srinadh Bhojanapalli, Daliang Li, Ankit Singh Rawat, Sashank J. Reddi, Ke Ye, Felix Chern, Felix X. Yu, Ruiqi Guo, and Sanjiv Kumar. Large models are parsimonious learners: Activation sparsity in trained transformers. *CoRR*, abs/2210.06313, 2022.

[17] Dario Lazzaro, Antonio Emanuele Cinà, Maura Pintor, Ambra Demontis, Battista Biggio, Fabio Roli, and Marcello Pelillo. Minimizing energy consumption of deep learning models by energy-aware training. *arXiv preprint arXiv:2307.00368*, 2023.

[18] Sarada Krithivasan, Sanchari Sen, and Anand Raghunathan. Adversarial sparsity attacks on deep neural networks. *ArXiv*, abs/2006.08020, 2020.

[19] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics.

[20] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *ArXiv*, abs/1606.05250, 2016.

[21] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[22] Ilia Shumailov, Yiren Zhao, Daniel Bates, Nicolas Papernot, Robert Mullins, and Ross Anderson. Sponge examples: Energy-latency attacks on neural networks. In *2021 IEEE European symposium on security and privacy (EuroS&P)*, pages 212–231. IEEE, 2021.