From PEFT to DEFT: 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
bharatrunwal@gmail.com, tejaswinip@us.ibm.com, pin-yu.chen@ibm.com

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 multilayer perceptron (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. We demonstrate the effectiveness of our approach by utilizing mainstream PEFT techniques, including QLoRA, LoRA, Adapter, and Prompt/Prefix Tuning, to facilitate efficient model adaptation across diverse downstream tasks. Experiments show that our proposed method, **DEFT** (Density-Efficient Fine-Tuning), can consistently reduce activation density by up to **44.94%** on RoBERTa$_{\text{Large}}$ and by **53.19%** (encoder density) and **90.60%** (decoder density) on Flan-T5$_{\text{XXL}}$ (**11B**) compared to PEFT, using GLUE and QA (SQuAD) benchmarks respectively, while maintaining competitive performance on downstream tasks. We also introduce **ADA-DEFT**, an adaptive variant of our DEFT approach, which achieves significant memory and runtime savings during inference for large models. For instance, ADA-DEFT reduces runtime by **8.75%** and memory usage by **16.78%** in Flan-T5$_{\text{XL}}$, and by **2.79%** and **2.54%** respectively in Flan-T5$_{\text{XXL}}$. Additionally, we showcase that DEFT works complementarily with quantized and pruned models.

Code — https://github.com/IBM/DEFT

Introduction

With the advent of pre-trained large language models (LLMs) (Devlin et al. 2019; Radford et al. 2019; Raffel et al. 2020), fine-tuning (Howard and Ruder 2018) these models to adapt to a task has become prevalent. However, training these models and performing inference on them requires a significant amount of time, energy, and memory, thereby resulting in an enormous carbon footprint (Strubell, Ganesh, and McCallum 2019). Some methods to achieve faster and greener inference are by pruning the model parameters (Lee, Ajanthan, and Torr 2019; Tanaka et al. 2020), pruning the number of heads by analyzing the attention patterns (Behnke and Heafield 2020; Voita et al. 2019; Michel, Levy, and Neubig 2019), distilling the larger model to a smaller model (Sanh et al. 2019), quantizing the models to convert the weights to a lower precision (Dettmers et al. 2022; Zadeh et al. 2020), using mixture of experts (MOE) (Kudugunta et al. 2021; Rajbhandari et al. 2022), etc. In contrast, this paper focuses on accelerating the model inference by increasing the activation sparsity in the model. This is achieved by including a penalty for high activation density in the loss function.

Recent studies (Zhang et al. 2021; Li et al. 2022) have shown that in a transformer architecture, specifically in the intermediate outputs of MLP (Multi-Layer Perceptron) blocks with ReLU activations, only a fraction of neurons are activated for a given input, leading to the sparse activation maps as the output. Building upon this observation, we propose a novel density loss that encourages higher activation sparsity in pre-trained models when adapting to downstream tasks, effectively reducing the activation density.

Moreover, the induction of higher activation sparsity holds promising prospects for substantial energy savings, especially on modern hardware acceleration architectures like ASICs (Application Specific Integrated Circuits) (Lazzaro et al. 2023), which leverage zero-skip operations. By promoting sparsity in the activation maps of transformers, hardware can take advantage of zero-skip operations, skipping unnecessary computations on zero-valued activations, resulting in reduced power consumption and more efficient model inference. This energy-efficient approach becomes particularly advantageous for resource-constrained environments or applications with strict energy constraints.

In this work, we present Density-Efficient Fine-Tuning (DEFT) and its variant ADA-DEFT (Adaptive-DEFT), which induces activation sparsity using parameter-efficient fine-tuning (PEFT) techniques. We illustrate DEFT in Figure (1a). The bar plot shows the reduction in Activation Density (%), which is defined as the number of non-zero values in the intermediate output of MLP layers in transformer blocks averaged over the full validation set. Our proposed DEFT significantly lowers the activation density compared

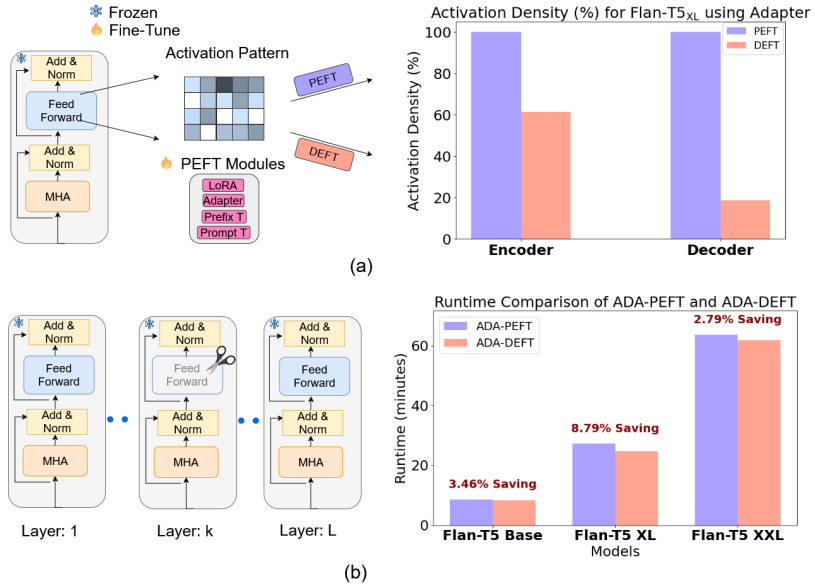


Figure 1: (a) Comparison between the activation density (in the intermediate output of MLP) after adapting to downstream tasks with PEFT and our proposed DEFT method. Both methods use Adapter. (b) ADA-DEFT during inference: based on the learned adaptive layerwise weights, we skip MLP blocks in the ADA-DEFT model, resulting in runtime savings for Flan-T5 models.

to PEFT. Figure (1b) illustrates our Adaptive DEFT (ADA-DEFT) method, where we skip the MLP block during inference based on the learned adaptive layerwise weights, resulting in runtime savings as shown in the bar plot for Flan-T5 models.

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 Gaussian Error Linear Unit (GeLU) (Hendrycks and Gimpel 2016) models (GeLU and it's variant are the default activation function of state-of-the-art transformer models). Prior studies primarily concentrated on ReLU-based models for investigating activation sparsity (Lazzaro et al. 2023), which are known for their inherent sparsity in activation maps. Our approach, combining PEFT and activation sparsity, paves the way for resource-friendly transformer models across various applications.

Related Work

Weight Induced Sparsity

Reducing the number of model parameters results in a model with a lesser memory footprint, reducing the amount of computational resources required to perform the inference. This can be done by pruning those model parameters whose removal does not deteriorate the model's performance significantly. To prune the weights at initialization using unstructured pruning, methods such as snip (Lee, Ajanthan, and Torr 2019), grasp (Wang, Zhang, and Grosse 2020), synaptic flow (Tanaka et al. 2020), etc can be leveraged.

Prior arts such as (Li, Cotterell, and Sachan 2021; Voita et al. 2019; Michel, Levy, and Neubig 2019; Behnke and Heafield 2020) sort the heads based on various importance scores and prune the bottom rung. For instance, in (Michel, Levy, and Neubig 2019), the importance score is the difference in loss values of when the head is not pruned and when it is pruned. Both (Voita et al. 2019) and (Behnke and Heafield 2020) use the confidence of heads as the pruning metric. Distilling the original model into a smaller model with fewer parameters also achieves the same goal. DistilBert (Sanh et al. 2019) was obtained by using knowledge distillation on the BERT model during the pre-training phase and is 40% smaller than BERT while being 97% as performant as BERT.

(Chen et al. 2021) used a parameter efficient fine-tuning technique that performs low rank weight updates similar to LoRA. The user inputs the desired sparsity and whether entire heads must be pruned or any model parameters can be pruned. After training, the model parameters and the heads are sorted by the gradient magnitude and are pruned according to the user's preference to achieve the desired sparsity. In (Sun et al. 2023a), the importance score for pruning model parameters is calculated as the product of the weight magnitudes and the norms of the input activations.

Activation Induced Sparsity

Rather than eliminating the parameters apriori, inducing activation sparsity dynamically reduces the latency on a sparsity-aware hardware by reducing the number of computations. (Li et al. 2022) showed that the larger the language models, the sparser their layer outputs are. Although the output was sparse, there was never a neuron that was never activated. While 93.5% of the neurons were activated less than 10% of the time, the least activated neuron was fired at least 0.001% of the time. Activation sparsity is achieved by thresholding the top-k activation outputs and zeroing out the

rest.

(Kurtz et al. 2020) modified ReLU activation of ResNet18 models to fire only if the magnitude of the input is higher than the specified threshold. A special sparsity-aware convolution algorithm is used to accelerate inference. As opposed to this, our method naturally induces the activation sparsity during the training owing to our loss function.

Methodology

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 (Li et al. 2022) and (Zhang et al. 2021).

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.

Gated-MLP Blocks: Most large language models (LLMs) currently use the Gated MLP blocks (Shazeer 2020). The Gating Mechanism consists of the following computations :

$$Y = \left(f(XW_s^T) \odot (XW_e^T)\right) W_o^T \tag{2}$$

Here $W_s, W_e \in \mathbb{R}^{d_{\text{ff}} \times d_{\text{model}}}$ and $W_o \in \mathbb{R}^{d_{\text{model}} \times d_{\text{ff}}}$.

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

$$O = f\left(XW_1\right) \text{ or } O = f\left(XW_s^T\right) \tag{3}$$

The matrix $O \in \mathbb{R}^{B \times K \times d_{\text{ff}}}$ is the activation pattern. Following (Li et al. 2022), 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.

Density Loss

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 (Li et al. 2022) 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 as defined in (4) for an input feature x with n elements $\{x_i\}_{i=1}^n$. (Krithivasan, Sen, and Raghunathan 2020) used a similar function for their purpose of generating adversarial inputs for sparsity attacks. We also use a differentiable approximation of the l_0 norm (Lazzaro et al. 2023) for GeLU and other models with activations different from GeLU and ReLU, as defined (5) with some hyperparameter $\epsilon > 0$.

$$\tanh(x, \beta) = \frac{e^{\beta \cdot x} - e^{-\beta \cdot x}}{e^{\beta \cdot x} + e^{-\beta \cdot x}} \tag{4}$$

$$\hat{l}_0(x, \epsilon) = \sum_{i=1}^{n} \left(\frac{x_i^2}{x_i^2 + \epsilon} \right) \tag{5}$$

By adjusting the value of β in (4), 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 (5), $\epsilon \in \mathbb{R}$ is a parameter dictating the quality of the approximation—lower values of ϵ correspond to better approximations. In Appendix A.3, we also explored different approximation functions like sigmoid, l_1 norm and others.

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

$$\mathcal{L}_{\text{density}}(x) = \frac{1}{n} \sum_{l=1}^{L} \sum_{i} g(s_{l_i}) \tag{6}$$

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 feature map after the first dense layer in MLP layer l, with i indexing each feature of s_l. The summation is across all the layers and the elements of the vector s_l. The approximation function g can either be the l_0 approximation (5) or any function in Appendix A.3.

Parameter Efficient Fine-tuning (PEFT)

Fine-tuning a LLM is computationally expensive, as it involves training all the model parameters from scratch. When adapting a model to multiple datasets, the traditional fine-tuning approach necessitates saving all the trained parameters separately for each dataset, leading to significant storage overhead and computational burden. PEFT addresses this by introducing fewer additional trainable parameters. During fine-tuning, only these parameters are trained, with the rest of the model remaining unchanged. This approach not only reduces computational burden but also minimizes storage demands, as only the new parameters need saving, streamlining the model adaptation process.

Our proposed DEFT is fully compatible with PEFT, and in this paper it adopts Prompt tuning (Lester, Al-Rfou, and Constant 2021), Prefix tuning (Li and Liang 2021), adapters (Houlsby et al. 2019) and Low-rank adaptation (LoRA) (Hu et al. 2022; Dettmers et al. 2023) techniques for demonstration. Each of these techniques is detailed in Appendix A.1.

DEFT: Parameter and Activation Density Efficient Fine-tuning

In our DEFT framework, to efficiently adapt the model without fine-tuning all parameters, we freeze the transformer parameters and only train the aforementioned PEFT modules for downstream tasks.

For PEFT, we solve the following optimization problem:

$$\arg\min_{\Phi} \mathcal{L}_{\text{T}}\left(\text{D}; \{\Theta, \Phi\}\right) \tag{7}$$

Algorithm 1: DEFT

Require: Dataset D, # of Epochs E, Batch Size B, # of Transformer Blocks L, Tunable Parameters Φ, Coefficient α, Sparsity Approximation Function g, Adaptive Sparsity Weights for ADA-DEFT $\{S\}_{i=1}^{L}$

for epoch $\leftarrow 1$ to E **do**
  for batch $\leftarrow 1$ to length(D) with batch size B **do**
    auxiliary variable $\eta \leftarrow []$
    for i $\leftarrow 1$ to L **do**
      $O_i \leftarrow$ Get the output for MLP$_i$
      η.append($g(O_i)$)
      η.append($g(s_i \cdot O_i)$) # ADA-DEFT
    end for
    Density Loss: $\mathcal{L}_{\text{density}} \leftarrow \text{mean}(\eta)$
    Total Loss: $\mathcal{L}_{\text{total}} \leftarrow \mathcal{L}_T + \alpha \cdot \mathcal{L}_{\text{density}}$
    Update parameters Φ using $\mathcal{L}_{\text{total}}$ loss
    Update parameters Φ, S using $\mathcal{L}_{\text{total}}$ loss # ADA-DEFT
  end for
end for

Here, Φ represents a set of additional parameters (tunable) in PEFT, while Θ denotes the set of pre-trained parameters (frozen). The loss function $\mathcal{L}_{\text{T}}$ encapsulates the task-specific objectives and D is the dataset associated with the task.

For our proposed density-efficient fine-tuning (**DEFT**), i.e. inducing activation sparsity in the MLP layer of transformer blocks, we augment the optimization problem (7) by incorporating our density loss (6):

$$\begin{aligned}\arg\min_{\Phi} \mathcal{L}_{\text{total}} = \mathcal{L}_{\text{T}}\left(\text{D}; \{\Theta, \Phi\}\right) \\ + \alpha \cdot \mathcal{L}_{\text{density}}\left(\text{D}; \{\Theta, \Phi\}\right)\end{aligned} \tag{8}$$

Here, the parameter α controls the balance between optimizing the performance metric and inducing activation sparsity. Notably, higher values of α promote sparser activation maps, although a careful equilibrium is required to ensure minimal impact on performance. We have described the algorithm for DEFT in Algorithm 1.

Importantly, the tunable parameters Φ encompass a versatile range of modules, or compositions thereof, from the choice of {Adapters, LoRA, QLoRA, Prefix-Tuning, Prompt-Tuning}, while the original pre-trained parameters Θ remains frozen. We advocate for these parameter-efficient modules over full-finetuning due to our experimental findings, which reveal that the introduction of a small fraction of trainable parameters (just a few % of the full model size) suffices to trigger activation sparsity within the MLP blocks. By incorporating these modules, we achieve a twofold efficiency advantage: (1) Facilitating activation sparsity, primed for utilization by hardware accelerators such as ASIC; and (2) Efficient training and storage of these modules, yielding gains in both training time and memory utilization, all while preserving the integrity of downstream task performance.

ADA-DEFT : Adaptive Parameter and Activation Density-Efficient Fine-Tuning

In previous section, we introduced an additional density loss to induce activation sparsity in pretrained models, with the weight of the loss being a constant α, which determines the sparsity in the activations. From prior work in weight pruning (Frankle and Carbin 2018; Mocanu et al. 2017), it has been observed that performance improves when the sparsity ratio allocation is non-uniform, i.e., each layer is treated differently for the downstream task. Inspired by the non-uniform layerwise weight sparsity, we investigated this non-uniform treatment of activation sparsity for each layer by introducing an extra trainable parameter for each MLP block in the model, formally:

$$\begin{aligned}\arg\min_{\Phi,\text{S}} \mathcal{L}_{\text{total}} = \mathcal{L}_{\text{T}}\left(\text{D}; \{\Theta, \Phi, \text{S}\}\right) \\ + \alpha \cdot \mathcal{L}_{\text{density}}\left(\text{D}; \{\Theta, \Phi, \text{S}\}\right)\end{aligned} \tag{9}$$

Here, $S = [S_1, S_2, \ldots, S_L]$ with $S_k \in [0, 1]$ for every $k \in [1, L]$ are the trainable parameters for each MLP Block (adaptive layerwise weights).

In our experiments, we demonstrate that the adaptive layerwise weights help skip some unimportant layers, resulting in memory and runtime savings with minimal impact on downstream performance.

Experiments

Datasets: We evaluated the performance of our method and PEFT techniques using two benchmark datasets: GLUE (Wang et al. 2018) and SQuAD (Rajpurkar et al. 2016). GLUE includes various natural language processing tasks. We focused on eight specific datasets: sentiment classification (SST-2), paraphrase detection (MRPC, QQP), natural language inference (MNLI, RTE, QNLI), linguistic acceptability (CoLA), and Semantic Textual Similarity (STS-B). SQuAD is a well-known reading comprehension benchmark. It comprises of question-answering pairs, requiring the model to provide answers based on given passages. More details about the datasets are in Appendix A.2.

Pretrained Language Models: In the main paper, we used pre-trained RoBERTa$_{\text{Large}}$ (355M parameters, 24 layers) (Liu et al. 2019); T5$_{\text{SMALL}}$ (60M parameters; 6 encoder and decoder layers), T5$_{\text{BASE}}$ (220M parameters; 12 encoder and decoder layers) (Raffel et al. 2019) models; Flan-T5-base (250M parameters, 12 encoder and decoder layers), Flan-T5-xl (3B parameters; 24 encoder and decoder layers), Flan-T5-xxl (11B parameters; 24 encoder and decoder layers) (Chung et al. 2022) instruction-tuned models. We also provide additional results with other models, including BERT, OPT, GPT2, and ViT in Appendix A.7.

PEFT Modules: We used {Adapter, LoRA, Prefix-Tuning (Prefix-T), Prompt Tuning (Prompt-T)} for {RoBERTa} and {Adapter, LoRA, QLoRA} for T5 models. These PEFT modules serve as the baselines to be compared with our proposed DEFT method. More detailed information about the hyperparameters employed in our experiments can be found in Appendix A.2.

Evaluation Metrics: Performance Regarding the GLUE benchmark, we utilize task-specific evaluation metrics. For the Semantic Textual Similarity (STS-B) dataset, we report the Pearson correlation coefficient. For the CoLA dataset, we use the Matthews correlation coefficient. For MNLI, we report accuracy on the matched validation set, while for all other GLUE tasks, we report accuracy. For the SQuAD dataset, we use the F1 score and Exact-Match score to evaluate performance.

Evaluation Metrics: Efficiency Beyond task-specific metrics, we evaluate the effectiveness of our method in promoting activation sparsity. We calculate the **Density** (%) of activations by identifying the number of non-zero values in the intermediate activation matrices within the MLP block of each transformer layer and averaging these across all layers and the validation set.

We also introduce the **Density Change** (%) metric, inspired by the energy consumption ratio concept from (Shumailov et al. 2021). This metric compares the sparsity induced by our method to the baseline and is computed as follows:

$$\text{Density Change (\%)} = \left(\frac{\text{Density}_{\text{PEFT}} - \text{Density}_{\text{DEFT}}}{\text{Density}_{\text{PEFT}}} \right) \times 100 \tag{10}$$

Here, $\text{Density}_{\text{PEFT}}$ and $\text{Density}_{\text{DEFT}}$, represent the density percentages for baseline PEFT and our DEFT methods, respectively. This formula effectively highlights the reduction in activation density achieved through our approach.

Energy Consumption Ratio Activation sparsity can be directly leveraged on hardware with zero-skip operations, such as ASIC accelerators. Thus, our DEFT method, which promotes sparser activations, is expected to yield higher energy savings on such hardware. We utilize the energy consumption ratio from (Lazzaro et al. 2023), defined as the ratio between the energy consumed with zero-skipping operations and the energy consumed with standard operations (without zero-skipping). We also report **Energy Change** (%), calculated as the relative change between the energy ratios of PEFT and DEFT, normalized by the PEFT energy ratio.

Runtime and Memory Analysis We report runtime in seconds and memory (in GB) usage for both ADA-DEFT and the baseline ADA-PEFT during evaluation, showcasing practical speedups in inference and reductions in memory storage with ADA-DEFT. This approach provides a direct comparison of real-world performance efficiencies.

Density Loss Hyperparameters: For our density loss we used $\beta = 20$ with tanh approximation Eq. (4) and $\epsilon = 1e-07$ with l_0-approximation Eq. (5) and $\alpha = 1.0$. We also provide an ablation study for varying these parameters later in Appendix A.5. We initialize the adaptive layerwise weights in Eq. 9 for both the encoder and decoder to 0.80 and then perturbed by adding random noise drawn from a normal distribution with a standard deviation of 0.05.

Results on GLUE Benchmark

The performance comparison of different methods on GLUE using the $\text{RoBERTa}_{\text{Large}}$ model is presented in Table **??**. This table provides insights into the effects of induced activation sparsity on both performance metrics and activation density in the intermediate layers of the Transformer MLP block with only a few trainable parameters.

Across all datasets, the fine-tuning methods, Adapter, LoRA, Prefix-Tuning (Prefix-T), and Prompt-Tuning (Prompt-T), were evaluated using two approaches: PEFT and DEFT. We observe that all DEFT methods (Adapter, LoRA, Prefix-T, and Prompt-T) generally achieve comparable or better performance to PEFT, with only marginal differences in most cases, while significantly reducing activation density. The best performance on the GLUE benchmark is observed with DEFT using the Adapter module (**88.06%**). The results consistently show that all DEFT methods achieve significantly lower activation density than PEFT.

Reduction in Activation Density: Adapter > LoRA > Prefix-T > Prompt-T. In all cases, our proposed DEFT method promotes activation sparsity with minimal or no effect on downstream performance. The reductions in activation density range from 0.02% (Prompt-T, MRPC) to 55.57% (Adapter, SST-2) across the different datasets and methods. Notably, the method achieving the highest reduction in activation density on the GLUE benchmark is Adapter (44.94%), followed by LoRA (38.82%) and Prefix-T (36.39%). Prompt-T shows the least reduction, at 1.44%, and we specifically note a training collapse in the CoLA dataset using Prompt-Tuning.

It is important to note that these results are not directly comparable across different modules due to variations in the number of trainable parameters and the locations of additional parameters. For instance, prefix-tuning involves adding trainable parameters to the hidden states. Nonetheless, the generality of DEFT is evident, as all methods effectively promote activation sparsity with minimal impact on downstream performance.

Layerwise Activation Sparsity Analysis. To delve deeper into the effects of our method, we analyze layerwise activation sparsity in $\text{RoBERTa}_{\text{Large}}$ when paired with Adapter. This analysis is visually represented in Fig. 2 for the SST-2 (a) and MNLI (d) datasets. Given that $\text{RoBERTa}_{\text{Large}}$ comprises 24 layers, we computed the percentage of non-zero activations across these layers using the validation datasets. The resulting plots reveal a pronounced decrease in non-zero activations at each layer, underscoring DEFT's efficiency in inducing activation sparsity throughout the network's depth.

Energy Consumption Ratio for $\text{RoBERTa}_{\text{Large}}$. In Table 2, we report the energy consumption ratio and energy change (%). We used SST2 dataset from GLUE benchmark and reported our results on **$\text{RoBERTa}_{\text{Large}}$** with Adapter module, the remaining modules are reported in Appendix A.6. From the results, we can see that DEFT leads to a reduction in Energy Consumption (8.23%) compared to PEFT

Module (% Trainable)	Method	Performance	MNLI	QQP	QNLI	SST-2	STS-B	MRPC	RTE	CoLA	Avg.
Adapter (1.17%)	PEFT	Metric (↑)	89.83	91.79	94.49	96.06	92.31	89.29	84.11	65.43	87.91
		Density (↓)	94.24	94.06	94.23	93.83	94.41	94.32	94.54	94.19	-
	DEFT	Metric (↑)	89.76	91.32	93.67	96.17	91.76	89.62	85.08	67.14	**88.06**
		Density (↓)	44.29	42.38	46.60	41.50	59.85	63.47	75.15	42.01	-
		DC (%) (↑)	53.00	54.94	50.55	55.77	36.61	32.71	20.51	55.40	**44.94**
LoRA (1.16%)	PEFT	Metric (↑)	90.53	91.38	94.71	95.67	91.21	91.63	81.94	63.21	**87.54**
		Density (↓)	94.61	94.35	94.49	93.87	94.32	94.28	94.50	94.18	-
	DEFT	Metric (↑)	90.27	90.79	93.89	95.99	91.52	85.11	81.40	61.33	86.29
		Density (↓)	43.64	49.08	48.65	45.86	87.61	44.00	85.83	57.01	-
		DC (%) (↑)	53.87	47.98	48.51	51.15	7.11	53.33	9.17	39.47	38.82
Prefix-T (1.11%)	PEFT	Metric (↑)	89.99	89.77	94.59	95.64	90.52	86.76	74.84	59.10	**85.15**
		Density (↓)	94.88	94.31	94.14	94.20	94.24	93.87	94.14	93.73	-
	DEFT	Metric (↑)	89.89	89.53	94.40	95.72	90.58	87.66	71.60	61.35	85.09
		Density (↓)	50.83	46.16	56.19	51.91	74.53	71.88	76.51	51.18	-
		DC (%) (↑)	46.43	51.06	40.31	44.89	20.91	23.43	18.73	45.4	36.39
Prompt-T (0.31%)	PEFT	Metric (↑)	81.53	84.17	80.88	84.63	22.66	71.159	51.98	2.18*	59.89
		Density (↓)	93.78	93.74	93.10	93.64	93.77	93.67	93.82	93.55	-
	DEFT	Metric (↑)	81.86	83.93	81.34	84.51	24.67	71.07	51.98	0.00*	**59.92**
		Density (↓)	88.84	89.29	93.00	92.90	93.74	93.65	93.79	93.05	-
		DC (%) (↑)	5.27	4.75	0.11	0.79	0.03	0.02	0.03	0.53	1.44

Table 1: Performance comparison on GLUE benchmarks with $\text{RoBERTa}_{\text{large}}$. (*) denotes unstable training. DC(%) represents Density Change(%). We present the mean values here; please refer to the appendix for the standard deviation.

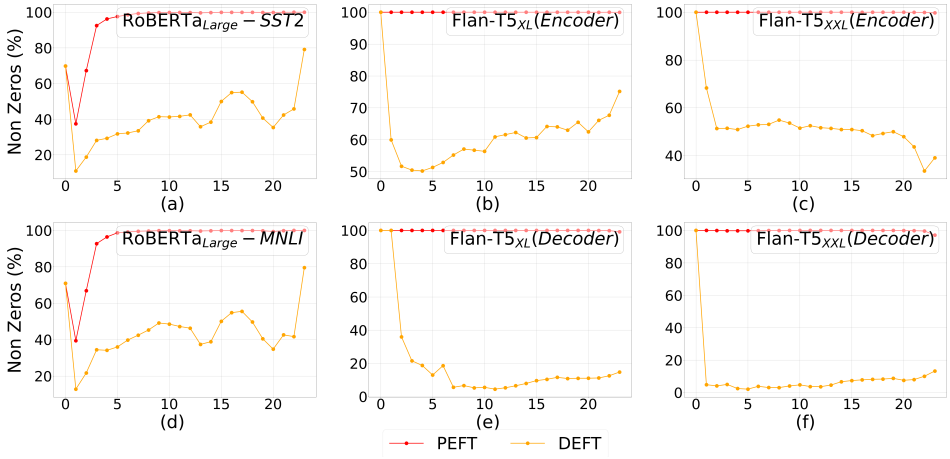


Figure 2: **Percentage of non-zeros (density).** Layerwise non-zeros (%) for $\text{RoBERTa}_{\text{Large}}$ (a,d), $\text{Flan-T5}_{\text{xl}}$ (b,e) with Adapter, and $\text{Flan-T5}_{\text{XXL}}$ (c.f) with QLoRA on the validation set of different tasks. The x-axis is the layer index.

on the ASIC simulator.

Results on SQuAD Dataset

Table 3 offers a comparative analysis of various methods applied to SQuAD using four T5 models. The evaluation focuses on two performance metrics: F1 score and Exact-Match, which measure the accuracy of the model's answers. Additionally, the table provides information on the encoder and decoder activation densities. In our experiments, we introduce the density loss for both encoder and decoder layers with an equal weightage of 1.0, though introducing the density loss for only one of them is also possible. T5_{Small} and T5_{Base} use ReLU activations, while $\text{Flan-T5}_{\text{XL}}$ and $\text{Flan-T5}_{\text{XXL}}$ use Gated FFN blocks with GeLU-based activation.

Larger models introduce more sparse activation patterns with DEFT. For ReLU-based models (T5_{Small} and

Model	Method	ER(↓)
$\text{RoBERTa}_{\text{Large}}$ (Adapter)	PEFT	0.85
	DEFT	0.78
	Energy Change(%)	**8.23%**
$\text{Flan-T5}_{\text{XL}}$ (Adapter)	PEFT	1.00
	DEFT	0.87
	Energy Change(%)	**13%**
$\text{Flan-T5}_{\text{XXL}}$ (QLoRA)	PEFT	1.00
	DEFT	0.85
	Energy Change(%)	**15%**

Table 2: Energy Consumption Ratio with different models using ASIC Simulator developed in (Shumailov et al. 2021).

Model	Module (% Trainable)	Loss Type	SQuAD			
			F1(↑)	EM(↑)	ED(↓)	DD(↓)
T5- Small	Adapter	PEFT	82.58	74.48	4.76	4.07
		DEFT	82.41	74.19	3.51	1.95
	(0.33%)	DC(%)			26.26	52.08
(60M)	LoRA	PEFT	82.60	74.54	4.80	3.97
		DEFT	82.38	74.19	3.33	1.51
	(0.96%)	DC(%)			30.62	61.96
T5- Base	Adapter	PEFT	88.28	81.19	2.64	3.22
		DEFT	88.21	81.08	1.61	0.96
	(0.40%)	DC(%)			39.01	70.19
(220M)	LoRA	PEFT	88.33	81.30	2.70	3.19
		DEFT	88.42	81.40	1.41	0.58
	(0.78%)	DC(%)			47.77	81.82
Flan-T5$_{\mathrm{XL}}$	Adapter	PEFT	92.81	87.28	99.99	99.96
		DEFT	92.52	86.79	61.53	18.70
(3B)	(0.87%)	DC(%)			38.46	81.29
Flan-T5$_{\mathrm{XXL}}$	QLoRA	PEFT	92.84	86.75	99.97	99.82
		DEFT	92.72	87.04	46.80	9.38
(11B)	(1.04%)	DC(%)			**53.19**	**90.60**

Table 3: Performance comparison of different methods on Question Answering Dataset (SQuAD) with T5 models. EM: Exact-Match, ED: Encoder Density ; DD: Decoder Density

T5$_{\mathrm{Base}}$), using the tanh-approximation Eq. (4) in the density loss, DEFT achieves comparable performance to PEFT with notable reductions in activation density. For T5$_{\mathrm{Small}}$, encoder density reductions range from 26.26% to 30.62% and decoder from 52.08% to 61.96%. For T5$_{\mathrm{BASE}}$, encoder reductions are 39.01% to 47.77% and decoder 70.19% to 81.82%, without sacrificing downstream performance.

For Flan-T5 models, which utilize GeLU activations, we used the l_0-approximation specified in Eq. (5) in the density loss of Eq. (6). We investigated our method on larger instruction-tuned models, namely Flan-T5$_{\mathrm{XL}}$ (3B) and Flan-T5$_{\mathrm{XXL}}$ (11B). From the results, our method consistently achieves comparable performance with PEFT while significantly reducing activation density for both encoder and decoder layers for larger models. For the Flan-T5$_{\mathrm{XL}}$ (3B) model with only 0.87% trainable parameters, we achieve density change (%) for the encoder **38.46%**, while for the decoder, we achieve **81.29%** when compared to PEFT methods. Similarly, for Flan-T5$_{\mathrm{XXL}}$ (11B) model, with only 1.04% trainable parameters, we achieve density change (%) of **53.19%** for the encoder and **90.60%** for Decoder compared to PEFT. These findings indicate that DEFT tends to induce sparser activation patterns as the model size increases.

Layerwise Activation Sparsity Analysis. To provide further insights, we present layerwise non-zero (%) activation plots in Fig. 2 using Flan-T5 models on SQuAD. Both models have 24 layers for both encoder and decoder. The plots distinctly show that DEFT significantly reduces the number of non-zero activations in both the encoder (b,c) and decoder (e,f) layers. A notable observation is the more pronounced reduction in non-zeros in the decoder layers as compared to the encoder layers.

Model	Module (% Trainable)	Loss Type	SQuAD			
			F1 (↑)	EM(↑)	Runtime (s)(↓)	Memory (GB)(↓)
Flan-T5$_{\mathrm{BASE}}$	LoRA	ADA-PEFT	89.60	82.85	511.83	0.95
		ADA-DEFT	89.50	82.60	494.13	0.88
(250M)	(2.67%)	Saving (%)			**3.46**	**7.37**
Flan-T5$_{\mathrm{XL}}$	QLoRA	ADA-PEFT	93.11	87.56	1632.89	2.92
		ADA-DEFT	92.30	86.56	1489.28	2.41
(3B)	(1.99%)	Saving(%)			**8.79**	**17.46**
Flan-T5$_{\mathrm{XXL}}$	QLoRA	ADA-PEFT	92.74	86.50	3815.67	10.61
		ADA-DEFT	92.57	86.89	3709.23	10.34
(11B)	(1.04%)	Saving(%)			**2.79**	**2.54**

Table 4: Performance comparison of **ADA-DEFT** on Question Answering Dataset (SQuAD) with Flan-T5 models.

Energy Consumption Ratio for Flan-T5 models. In Table 2, we report energy consumption ratio and energy change (%) for Flan-T5 models. We can observe that DEFT leads to a decrease in energy consumption, especially for Flan-T5$_{\mathrm{XXL}}$, which demonstrates a noteworthy **15%** decrease in energy consumption with DEFT compared to PEFT.

ADA-DEFT with Flan-T5 models. We compare the ADA-DEFT method with the baseline ADA-PEFT across various configurations of the Flan-T5 model on the SQuAD dataset in Table 4. ADA-DEFT achieves comparable F1 and Exact Match (EM) scores to ADA-PEFT while offering substantial runtime and memory reductions. For the Flan-T5$_{\mathrm{BASE}}$, ADA-DEFT achieves an F1 score of 89.50 and an EM score of 82.60, with a **3.46%** runtime and **7.37%** memory savings. For the Flan-T5$_{\mathrm{XL}}$, it records a **8.79%** runtime and **17.46%** memory savings, and for the Flan-T5$_{\mathrm{XXL}}$, the savings are **2.79%** in runtime and **2.54%** in memory. These efficiencies highlight ADA-DEFT's capability to reduce resource usage without sacrificing performance. The final learned adaptive layerwise weights for both the encoder and decoder of all Flan-T5 models are shown in Fig. 3a. From the layerwise plots, we observe that for some of the MLP blocks, ADA-DEFT sets the adaptive layer weight to 0, allowing us to skip the MLP block during inference. This results in runtime and memory savings.

In summary, the results show that the Adapter and LoRA modules can maintain competitive performance on SQuAD while achieving notable reductions in activation density and runtime.

Pruning of PEFT/DEFT Models

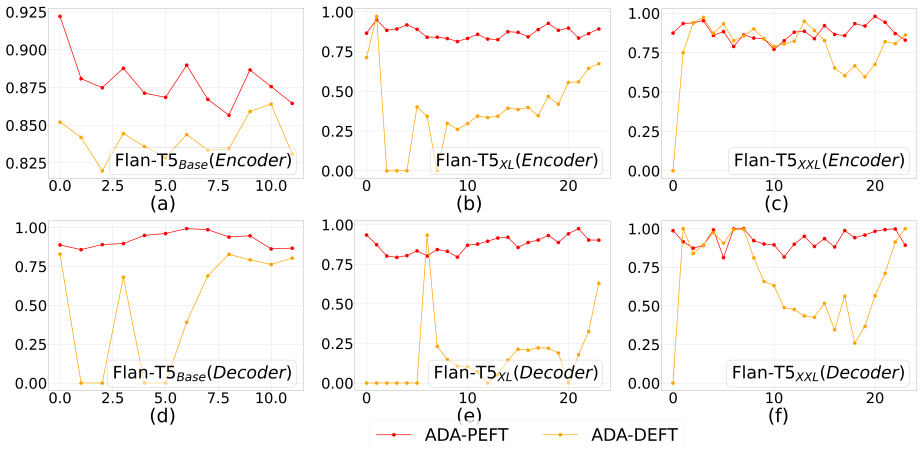
Here, we explore the effects of model pruning using the WANDA metric (Sun et al. 2023b). The WANDA metric combines weight magnitude and activations to identify parameters for pruning.

Given a weight matrix $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ and input activations $X \in \mathbb{R}^{N \times K \times d_{\text{in}}}$, the importance score of each weight is calculated as:

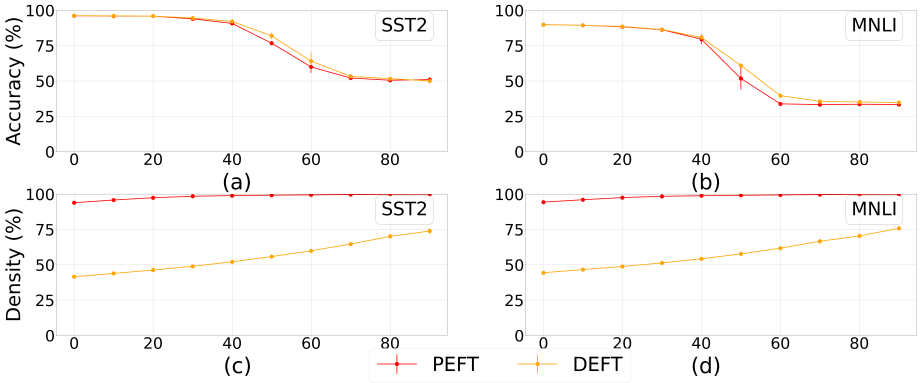
$$I_{ij} = |W_{ij}| \cdot ||X_j||_2 \tag{11}$$

where $||X_j||_2$ is the l_2 norm across the $N \times K$ tokens of the jth feature.

We randomly select 128 samples from the training set of each downstream dataset and use the WANDA metric to



(a) **Adaptive Layerwise Weights.** Learned Adaptive layerwise weights for ADA-DEFT and ADA-PEFT on SQuAD dataset using Flan-T5 models.



(b) **Metric v/s Sparsity.** Performance of RoBERTa$_{\text{Large}}$ with Adapter for different pruning thresholds for MLP block using WANDA metric on the validation set. (a) and (c): Accuracy and Density (%) on SST2 dataset. (b) and (d): Accuracy and Density (%) on MNLI dataset.

Figure 3: Comparison of Adaptive Layerwise Weights and Pruning Performance.

prune the first dense layer in the MLP blocks of transformer layers. The results are shown for models first adapted using PEFT/DEFT and then pruned using WANDA.

Fig.3b shows the performance of the pruned RoBERTa$_{\text{Large}}$ model on the SST-2 and MNLI validation sets. Initially, as sparsity increases, performance remains stable, but beyond a threshold, accuracy declines. DEFT consistently achieves higher or comparable accuracy than PEFT at various sparsity levels while maintaining greater activation sparsity. For example, as shown in Fig. 3b (b,d), at 50% sparsity, the model utilizing DEFT attains an accuracy of 60.96% with an activation Density of 57.65%, outperforming PEFT, which achieves only 51.73% accuracy with a significantly higher Density of 99.09%. This outcome suggests that DEFT can effectively complement weight pruning methods like WANDA, enabling models to benefit from both activation and weight sparsity.

Conclusion

In this work, we presented our methods DEFT and ADA-DEFT, novel add-on modules to PEFT for inducing activation sparsity in MLP layers of frozen pre-trained transformer blocks. We demonstrate the effectiveness of DEFT for reducing the activation density without hurting downstream performance compared to PEFT by various experiments on GLUE and SQuAD 1.0 benchmark with different models, and with different parameter-efficient modules. Extensive experimental results confirm that our proposed methods provide new means for density-efficient PEFT of pre-trained language models to improve inference efficiency while maintaining similar model performance. We also showcase the effect of pruning with the DEFT models and find that DEFT can be used as a complementary method with weight pruning methods, leading to both activation and weight sparsity. We believe that our proposed DEFT opens a new avenue for density-efficient fine-tuning of pre-trained models.

Acknowledgments

The authors thank Ming-Hung Chen and I-Hsin Chung at IBM Research for their help and discussion.

References

Behnke, M.; and Heafield, K. 2020. Losing Heads in the Lottery: Pruning Transformer Attention in Neural Machine Translation. In Webber, B.; Cohn, T.; He, Y.; and Liu, Y., eds., *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, 2664–2674. Association for Computational Linguistics.

Chen, X.; Chen, T.; Cheng, Y.; Chen, W.; Wang, Z.; and Awadallah, A. H. 2021. DSEE: Dually Sparsity-embedded Efficient Tuning of Pre-trained Language Models. *CoRR*, abs/2111.00160.

Chung, H. W.; Hou, L.; Longpre, S.; Zoph, B.; Tay, Y.; Fedus, W.; Li, E.; Wang, X.; Dehghani, M.; Brahma, S.; Webson, A.; Gu, S. S.; Dai, Z.; Suzgun, M.; Chen, X.; Chowdhery, A.; Valter, D.; Narang, S.; Mishra, G.; Yu, A. W.; Zhao, V.; Huang, Y.; Dai, A. M.; Yu, H.; Petrov, S.; hsin Chi, E. H.; Dean, J.; Devlin, J.; Roberts, A.; Zhou, D.; Le, Q. V.; and Wei, J. 2022. Scaling Instruction-Finetuned Language Models. *ArXiv*, abs/2210.11416.

Dettmers, T.; Lewis, M.; Belkada, Y.; and Zettlemoyer, L. 2022. LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale. *CoRR*, abs/2208.07339.

Dettmers, T.; Pagnoni, A.; Holtzman, A.; and Zettlemoyer, L. 2023. QLoRA: Efficient Finetuning of Quantized LLMs. *ArXiv*, abs/2305.14314.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv*, abs/1810.04805.

Frankle, J.; and Carbin, M. 2018. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. *arXiv: Learning*.

Hendrycks, D.; and Gimpel, K. 2016. Gaussian Error Linear Units (GELUs). *arXiv: Learning*.

Houlsby, N.; Giurgiu, A.; Jastrzebski, S.; Morrone, B.; de Laroussilhe, Q.; Gesmundo, A.; Attariyan, M.; and Gelly, S. 2019. Parameter-Efficient Transfer Learning for NLP. In Chaudhuri, K.; and Salakhutdinov, R., eds., *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*, 2790–2799. PMLR.

Howard, J.; and Ruder, S. 2018. Universal Language Model Fine-tuning for Text Classification. In Gurevych, I.; and Miyao, Y., eds., *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, 328–339. Association for Computational Linguistics.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2022. 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.

Krithivasan, S.; Sen, S.; and Raghunathan, A. 2020. Adversarial Sparsity Attacks on Deep Neural Networks. *ArXiv*, abs/2006.08020.

Kudugunta, S.; Huang, Y.; Bapna, A.; Krikun, M.; Lepikhin, D.; Luong, M.; and Firat, O. 2021. Beyond Distillation: Task-level Mixture-of-Experts for Efficient Inference. In Moens, M.; Huang, X.; Specia, L.; and Yih, S. W., eds., *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, 3577–3599. Association for Computational Linguistics.

Kurtz, M.; Kopinsky, J.; Gelashvili, R.; Matveev, A.; Carr, J.; Goin, M.; Leiserson, W. M.; Moore, S.; Shavit, N.; and Alistarh, D. 2020. Inducing and Exploiting Activation Sparsity for Fast Inference on Deep Neural Networks. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, 5533–5543. PMLR.

Lazzaro, D.; Cinà, A. E.; Pintor, M.; Demontis, A.; Biggio, B.; Roli, F.; and Pelillo, M. 2023. Minimizing Energy Consumption of Deep Learning Models by Energy-Aware Training. *arXiv preprint arXiv:2307.00368*.

Lee, N.; Ajanthan, T.; and Torr, P. H. S. 2019. 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.

Lester, B.; Al-Rfou, R.; and Constant, N. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. In Moens, M.; Huang, X.; Specia, L.; and Yih, S. W., eds., *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, 3045–3059. Association for Computational Linguistics.

Li, J.; Cotterell, R.; and Sachan, M. 2021. Differentiable Subset Pruning of Transformer Heads. *Trans. Assoc. Comput. Linguistics*, 9: 1442–1459.

Li, X. L.; and Liang, P. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In Zong, C.; Xia, F.; Li, W.; and Navigli, R., eds., *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*, 4582–4597. Association for Computational Linguistics.

Li, Z.; You, C.; Bhojanapalli, S.; Li, D.; Rawat, A. S.; Reddi, S. J.; Ye, K.; Chern, F.; Yu, F. X.; Guo, R.; and Kumar, S. 2022. Large Models are Parsimonious Learners: Activation Sparsity in Trained Transformers. *CoRR*, abs/2210.06313.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *ArXiv*, abs/1907.11692.

Michel, P.; Levy, O.; and Neubig, G. 2019. Are Sixteen Heads Really Better than One? In Wallach, H. M.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E. B.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Infor-*

mation Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, 14014–14024.

Mocanu, D. C.; Mocanu, E.; Stone, P.; Nguyen, P. H.; Gibescu, M.; and Liotta, A. 2017. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature Communications*, 9.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language Models are Unsupervised Multitask Learners.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.*, 21: 140:1–140:67.

Raffel, C.; Shazeer, N. M.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2019. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *ArXiv*, abs/1910.10683.

Rajbhandari, S.; Li, C.; Yao, Z.; Zhang, M.; Aminabadi, R. Y.; Awan, A. A.; Rasley, J.; and He, Y. 2022. DeepSpeed-MoE: Advancing Mixture-of-Experts Inference and Training to Power Next-Generation AI Scale. In Chaudhuri, K.; Jegelka, S.; Song, L.; Szepesvári, C.; Niu, G.; and Sabato, S., eds., *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, 18332–18346. PMLR.

Rajpurkar, P.; Zhang, J.; Lopyrev, K.; and Liang, P. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. *ArXiv*, abs/1606.05250.

Sanh, V.; Debut, L.; Chaumond, J.; and Wolf, T. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

Shazeer, N. M. 2020. GLU Variants Improve Transformer. *ArXiv*, abs/2002.05202.

Shumailov, I.; Zhao, Y.; Bates, D.; Papernot, N.; Mullins, R.; and Anderson, R. 2021. Sponge examples: Energy-latency attacks on neural networks. In *2021 IEEE European symposium on security and privacy (EuroS&P)*, 212–231. IEEE.

Strubell, E.; Ganesh, A.; and McCallum, A. 2019. Energy and Policy Considerations for Deep Learning in NLP. In Korhonen, A.; Traum, D. R.; and Màrquez, L., eds., *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, 3645–3650. Association for Computational Linguistics.

Sun, M.; Liu, Z.; Bair, A.; and Kolter, J. Z. 2023a. A Simple and Effective Pruning Approach for Large Language Models. *CoRR*, abs/2306.11695.

Sun, M.; Liu, Z.; Bair, A.; and Kolter, J. Z. 2023b. A Simple and Effective Pruning Approach for Large Language Models. arXiv:2306.11695.

Tanaka, H.; Kunin, D.; Yamins, D. L. K.; and Ganguli, S. 2020. Pruning neural networks without any data by iteratively conserving synaptic flow. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Voita, E.; Talbot, D.; Moiseev, F.; Sennrich, R.; and Titov, I. 2019. Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned. In Korhonen, A.; Traum, D. R.; and Màrquez, L., eds., *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, 5797–5808. Association for Computational Linguistics.

Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. 2018. 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*, 353–355. Brussels, Belgium: Association for Computational Linguistics.

Wang, C.; Zhang, G.; and Grosse, R. B. 2020. Picking Winning Tickets Before Training by Preserving Gradient Flow. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Zadeh, A. H.; Edo, I.; Awad, O. M.; and Moshovos, A. 2020. 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*, 811–824. IEEE.

Zhang, Z.; Lin, Y.; Liu, Z.; Li, P.; Sun, M.; and Zhou, J. 2021. MoEfication: Transformer Feed-forward Layers are Mixtures of Experts. In *Findings*.