Sparsity-Accelerated Training for Large Language Models

Da Ma[§], **Lu Chen**[§†*], **Pengyu Wang**[§†], **Hongshen Xu**[§], **Hanqi Li**[§]
Liangtai Sun[§], **Su Zhu**[‡], **Shuai Fan**[‡], **Kai Yu**[§†*]
[§]X-LANCE Lab, Department of Computer Science and Engineering
MoE Key Lab of Artificial Intelligence, SJTU AI Institute
Shanghai Jiao Tong University, Shanghai, China
[†]Suzhou Laboratory, Suzhou, China
[‡]AISpeech Co., Ltd., Suzhou, China
{mada123, chenlusz, kai.yu}@sjtu.edu.cn

Abstract

Large language models (LLMs) have demonstrated proficiency across various natural language processing (NLP) tasks but often require additional training, such as continual pre-training and supervised fine-tuning. However, the costs associated with this, primarily due to their large parameter count, remain high. This paper proposes leveraging *sparsity* in pre-trained LLMs to expedite this training process. By observing sparsity in activated neurons during forward iterations, we identify the potential for computational speed-ups by excluding inactive neurons. We address associated challenges by extending existing neuron importance evaluation metrics and introducing a ladder omission rate scheduler. Our experiments on Llama-2 demonstrate that Sparsity-Accelerated Training (SAT) achieves comparable or superior performance to standard training while significantly accelerating the process. Specifically, SAT achieves a 45% throughput improvement in continual pre-training and saves 38% training time in supervised fine-tuning in practice. It offers a simple, hardware-agnostic, and easily deployable framework for additional LLM training. Our code is available at https://github.com/OpenDFM/SAT.

1 Introduction

Large language models (LLMs), such as GPT-4 (Achiam et al., 2023), Mistral (Jiang et al., 2023), and Llama-2 (Touvron et al., 2023), have demonstrated remarkable capabilities across numerous NLP tasks (Mao et al., 2023; Dillmann et al., 2024). In general, all these models are initially pre-trained on massive data by unsupervised learning. Furthermore, such models regularly necessitate additional training for two primary scenarios: 1) *continual pre-training* on new data as the pool of available pre-training data continuously expands (Gupta et al., 2023; Cui et al., 2023; Ke et al., 2023). 2) *supervised fine-tuing* (SFT) on labeled data to enhance the capacity of LLMs to follow human instructions (Ouyang et al., 2022; Dong et al., 2023; Ansell et al., 2024).

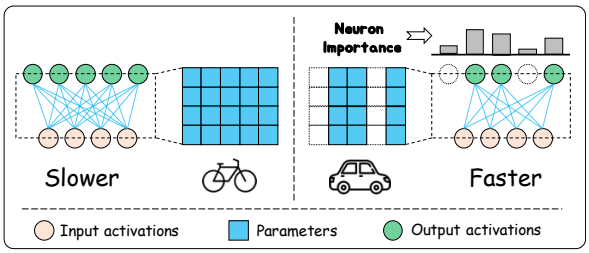


Figure 1: Insights into neuron sparsity in large language models to accelerate training. **Left**: standard linear layer. **Right**: the linear layer after pruning relatively less important neurons. By omitting computations related to some less important neurons, training on the right side proceeds faster compared to the left side, while maintaining similar performance.

In contrast to initial pre-training, the additional training demands relatively less time and computational resources. Regrettably, it remains somewhat challenging to afford, primarily attributable to the substantial parameter count (Zhao et al., 2024b; Wu et al., 2023). Fortunately, numerous practitioners (Zhao et al., 2024a) are dedicating efforts to saving such additional training costs. However, they mainly focus on the Parameter-Efficient Fine-Tuning (PEFT) methods (Ansell et al., 2024; Lialin et al., 2023), which is not commonly employed in the aforementioned continual pre-training (Cui et al., 2023) scenario. Hence, it is imperative to explore novel acceleration methods suitable for both additional training scenes simultaneously.

In this paper, our objective is to leverage the structural *sparsity* (Li et al., 2023b) inherent in pre-trained LLMs to expedite their additional training scenarios. In detail, researchers (Zhang et al., 2022b; Liu et al., 2023) have observed that the

*The corresponding authors are Lu Chen and Kai Yu.

Findings of the Association for Computational Linguistics ACL 2024, pages 14696–14707
August 11-16, 2024

activated (important) neurons[1] exhibit sparsity in each forward iteration. Intuitively, we can speed up by omitting the calculation of inactive (unimportant) neurons. As depicted in Figure 1, suppose a neuron corresponds to a particular column of the weight matrix in a linear layer. In each training iteration, a *compact* and *efficient* neuron importance assessment module (right part) would evaluate the activation status of neurons in advance. By discarding those inactive neurons (columns in the weight matrix), training speed is enhanced.

Aside from the enhancement in computational speed, disregarding the computation of inactive neurons also poses the following three additional challenges: 1) determining the method of neurons to discard, 2) assessing whether omitting neurons harms the model performance, and 3) evaluating the effectiveness of the approach in real additional training scenarios of LLMs. For the first challenge, we borrowed and extended several famous neuron importance evaluation metrics (Liu et al., 2023; Han et al., 2015; Sun et al., 2023) within the context of transformer pruning, and unimportant neurons are dropped. Extensive experiments (§ 3.1) on TinyLlama-1.1B (Zhang et al., 2024) help us determine (§ 2.2) 1) using maxip (a neuron importance metric) to assess neuron importance, 2) adopting sampling (versus top-k) neuron selection strategy, and 3) introducing a ladder omission rate scheduler.

For the latter two, we separately conduct experiments over Llama-2 (7B and 13B) in both standard and sparsity-accelerated manners for both continual pre-training and supervised fine-tuning scenarios (§ 3.2). Evaluation results on several benchmarks show that SAT achieves comparable sometimes even better performance compared to normal training. Moreover, it can obtain about 45% throughput improvement for continual pre-training and 38% speedup in elapsed time for supervised fine-tuning (§ 3.3). In summary, this work contributes in the following three aspects:

1. We propose the **S**parsity-**A**ccelerated **T**raining (SAT), a novel, hardware-agnostic, and easily deployable framework for additional training of LLMs.

2. We investigate several neuron importance computation and selection methods and summarize an optimal configuration for SAT. In addition, a novel ladder omission rate scheduler is designed to alleviate the overfitting problem of SAT.

3. Extensive experiments in both continual pre-training and supervised fine-tuning demonstrate that SAT achieves comparable performance to standard training of LLMs with speedup.

[1]In this paper, a neuron is associated with a specific row/column in a weight matrix.

2 Methodology

In this section, we first give an overview of SAT. Next, more details are provided within the context of transformers, which is the predominant architecture of recent LLMs. Finally, we discuss our implementation and conduct a theoretical efficiency analysis for the SAT.

2.1 Overview of SAT

Suppose a neural network model is expressed by $f(\cdot; \mathcal{N}_\theta)$, where $\mathcal{N}_\theta$ denotes the set of neurons. Recall that our objective is to disregard non-essential neurons to accelerate. Consequently, Sparse-Accelerated Training (SAT) can be delineated into the following two overarching steps for the t-th training iteration:

- *Neuron Importance Computation*: compute the importance scores for each neuron,

$$\mathbf{s}_\theta^t = \textit{ComputeImportance}\left(\cdot, \mathcal{N}_\theta^t\right), \quad (1)$$

  where $\mathbf{s}_\theta^t \in \mathbb{R}^{|\mathcal{N}_\theta^t|}$.

- *Neuron Selction and Optimization*: select important neurons upon $\mathbf{s}_\theta^t$ and update them,

$$\begin{aligned} \tilde{\mathcal{N}}_\theta^t =& \textit{SelectNeuron}\left(\mathbf{s}_\theta^t, r, \mathcal{N}_\theta^t\right), \\ \mathcal{N}_\theta^{t+1} =& \left\{ n_\theta \middle| n_\theta \in \left(\mathcal{N}_\theta^t - \tilde{\mathcal{N}}_\theta^t\right) \right. \\ & \left. \vee n_\theta \in \textit{Optimize}\left(\tilde{\mathcal{N}}_\theta^t\right) \right\}, \end{aligned} \quad (2)$$

  where $r \in [0, 1)$ denotes the neuron omission rate and $\tilde{\mathcal{N}}_\theta^t \subseteq \mathcal{N}_\theta^t$. Actually, we optimize the subnetwork expressed by $f(\cdot; \tilde{\mathcal{N}}_\theta^t)$ at training step t.

2.2 SAT for transformers

As widely recognized, contemporary LLMs are predominantly based on the Transformer (Vaswani et al., 2017) architecture. Therefore, our study primarily explores the SAT for transformers.

Review of Transformers The transformer is composed of numerous stacked layers, with each layer consisting of two main modules: Multi-Head Attention (MHA) and an immediately subsequent Multi-Layer Perceptron (MLP). Formally, let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n] \in \mathbb{R}^{n \times d}$ denote the input activations of the ℓ-th layer[2]. The MHA can be formulated as

$$\begin{aligned} \mathit{MHA}\left(\mathbf{X}\right) &= \mathit{Concat}\left(\mathit{head}_1, \ldots, \mathit{head}_h\right) \mathbf{W}_O, \\ \mathit{head}_i &= \mathbf{A}_i \cdot \left(\mathbf{X}\mathbf{W}_V^i\right), \\ \mathbf{A}_i &= \mathit{Softmax}\left(\frac{\left(\mathbf{X}\mathbf{W}_Q^i\right) \cdot \left(\mathbf{X}\mathbf{W}_K^i\right)^T}{\sqrt{d_k}}\right), \end{aligned} \tag{3}$$

where $h \in \mathbb{N}^+$ represents the number of heads and $d \in \mathbb{N}^+$ is the dimension of hidden states. $d_k = d/h$, $\mathbf{W}_Q^i, \mathbf{W}_K^i$ and $\mathbf{W}_V^i \in \mathbb{R}^{d \times d_k}$ $(1 \leq i \leq h)$, and $\mathbf{W}_O \in \mathbb{R}^{d \times d}$ are trainable parameters.

Subsequently, the MLP takes the output of MHA with a residual connection as input. Mathematically,

$$\begin{aligned} \mathbf{X} &= \mathbf{X} + \mathit{MHA}\left(\mathbf{X}\right), \\ \mathit{MLP}\left(\mathbf{X}\right) &= \sigma\left(\mathbf{X}\mathbf{W}_{up}\right) \cdot \mathbf{W}_{down}, \end{aligned} \tag{4}$$

where $\sigma\left(\cdot\right)$ is the activation function, $\mathbf{W}_{up} \in \mathbb{R}^{d \times 4d}$ and $\mathbf{W}_{down} \in \mathbb{R}^{4d \times d}$ are trainable parameters. The output of MLP with a residual connection is fed into the next layer[3]. Next, we shall proceed to a detailed discussion of the two steps of SAT within the Transformer architecture.

Neuron Importance Computation In this work, we leverage the structural sparsity of neurons across all linear layers, i.e., each row or column of the parameter matrix is conceptualized as a neuron. As our approach solely depends on the significance of columnar neurons, we exclusively focus on devising the methodology for assessing their importance. To clarify these methods, we first introduce some notations. Assume a column neuron $\mathbf{v} \in \mathbb{R}^{d \times 1}$ and it takes $\mathbf{Z} \in \mathbb{R}^{m \times d}$ as input. $\mathbf{Z}_i$ and $\mathbf{Z}^j$ are the i-th row and j-th column of $\mathbf{Z}$, respectively. $s_\mathbf{v}$ denotes the neuron importance score of $\mathbf{v}$. The activation value of $\mathbf{v}$ is $\mathbf{y} = \mathbf{Z} \cdot \mathbf{v}$. Next, we first mathematically define four evaluation metrics (Liu et al., 2023; Han et al., 2015; Sun et al., 2023) for neuron importance, which are

$$\begin{aligned} \mathit{uniform} &: s_\mathbf{v} = 1, \\ \mathit{magnitude} &: s_\mathbf{v} = \left\|\mathbf{v}\right\|_2, \\ \mathit{wanda} &: s_\mathbf{v} = \left[\left\|\mathbf{Z}^1\right\|_2, \ldots, \left\|\mathbf{Z}^d\right\|_2\right] \cdot \mathbf{v}, \\ \mathit{maxip} &: s_\mathbf{v} = \left(\sum_{i=1}^{m} \mathbf{Z}_i / m\right) \cdot \mathbf{v}. \end{aligned} \tag{5}$$

The *uniform* method is straightforward, as it entails randomly preserving neurons, akin to the well-known vanilla dropout technique (Hinton et al., 2012). The other three methods are predicated on the assumption that *neurons with larger activation values are relatively more important* (Liu et al., 2023). To elucidate this assumption, we can consider an example. In the multi-head self-attention mechanism of the Transformer, the representation of each token is weighted by the representations of all input tokens. At this juncture, the weights (attention scores) between tokens become pivotal. These attention scores stem from the dot product of the representations (activation values) of each token. The higher the activation value, the greater the probability of a larger dot product, yielding a higher attention score, thus indicating greater importance to the token representation.

Based on this assumption, the remaining three methods posit that neurons more likely to generate higher activation values ($\mathbf{y}$) are more important. In detail, the *magnitude* (Han et al., 2015) method solely focuses on the magnitude of the neuron parameter, i.e., the L2 norm of $\mathbf{v}$. The remaining two methods consider the influence of both input and parameters on the activation values. Concretely, *wanda* (Sun et al., 2023) considers the magnitude of the input and parameter product, while *maxip* (Liu et al., 2023; Song et al., 2023) focuses on the input mean and parameter product.

Subsequently, we will delve into separate discussions regarding neuron selection for MHA and MLP according to the neuron importance score.

Neuron Selection for MHA For MHA, we operate at the granularity of individual heads, where we either omit or retain all neurons within a head. For the i-th head, each of its three linear parameter matrices $\mathbf{W}_Q^i, \mathbf{W}_K^i, \mathbf{W}_V^i \in \mathbb{R}^{d \times d_k}$ includes d_k neurons, and the neuron importance score of the j-th neuron in each matrix is denoted as $s_Q^{i,j}$, $s_K^{i,j}$, $s_V^{i,j}$ respectively. We first calculate the importance

[2]To maintain manuscript tidiness, we omit the subscript of ℓ in this paper.

[3]We omit the writing of layer norm layer and attention mask here.

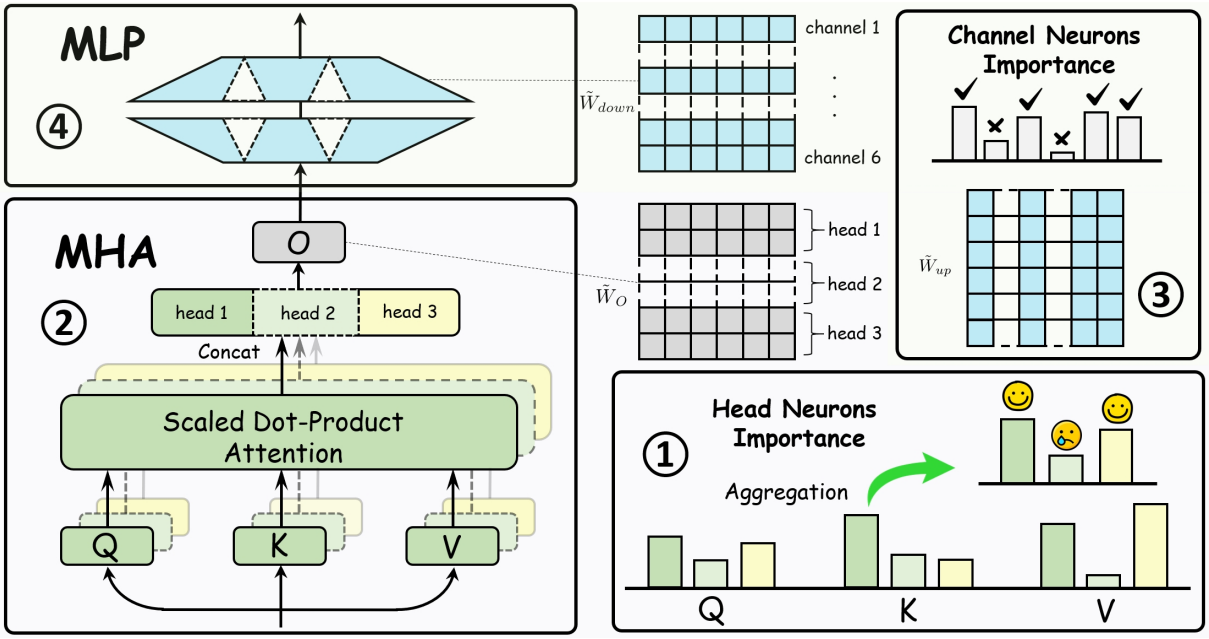


Figure 2: Sparsity-Accelerated Training (SAT) for transformers. ①: Process of selecting more important heads for MHA. ②: SAT for MHA. ③: Process of selecting more important channels for MLP. ④: SAT for MLP. Dashed parts denote omitted neurons.

score of the i-th head in each parameter matrix via

$$s_Q^{head_i} = \sum_{j=1}^{d_k} \frac{s_Q^{i,j}}{d_k}, s_K^{head_i} = \sum_{j=1}^{d_k} \frac{s_K^{i,j}}{d_k}, s_V^{head_i} = \sum_{j=1}^{d_k} \frac{s_V^{i,j}}{d_k}. \tag{6}$$

Then, the importance score of i-th head is aggregated, as depicted in Figure 2 (①) by

$$s^{head_i} = \left(s_Q^{head_i} + s_K^{head_i} + s_V^{head_i}\right) \Big/ 3. \tag{7}$$

Subsequently, $\tilde{h} = \lfloor h \times (1-r) \rfloor$ heads will be selected to optimize based on the head importance score in the following two ways:

- *top-k*: retain the top $\tilde{h}$ heads with the highest importance scores.
- *sampling*: sample $\tilde{h}$ heads according to the distribution:

$$head_i \sim p\left(head_i\right), \quad p\left(head_i\right) = \frac{exp\left(s^{head_i}/\tau\right)}{\sum_j exp\left(s^{head_j}/\tau\right)}, \tag{8}$$

  where $1 \leq i \leq h$ and $\tau \in \mathbb{R}^+$ is the temperature for sampling (Dupont et al., 2022).

As is shown in Figure 2 (②), only selected $\tilde{h}$ heads participate in the computation. Moreover, only the rows corresponding to the selected heads in $\mathbf{W}_O$ (referred to as $\tilde{\mathbf{W}}_O$) will be involved in the computation. Assuming heads $(\pi_1, \pi_2, \ldots, \pi_{\tilde{h}})$ are selected, the formula (3) becomes

$$MHA\left(\mathbf{X}\right) = Concat\left(head_{\pi_1}, \ldots, head_{\pi_{\tilde{h}}}\right) \tilde{\mathbf{W}}_O. \tag{9}$$

Neuron Selection for MLP For MLP, we operate at the granularity of individual channels, where each channel (column) in $\mathbf{W}_{up} \in \mathbb{R}^{d \times 4d}$ corresponds to a neuron and we select $\tilde{c} = \lfloor 4d \cdot (1-r) \rfloor$ neurons (denoted as $\tilde{\mathbf{W}}_{up} \in \mathbb{R}^{d \times \tilde{c}}$) to optimize similarly to MHA (see Figure 2 (③)). Likewise, rows corresponding to the selected channels in $\mathbf{W}_{down} \in \mathbb{R}^{4d \times d}$ (denoted as $\tilde{\mathbf{W}}_{down} \in \mathbb{R}^{\tilde{c} \times d}$) will participate in the computation as illustrated in Figure 2 (④). Then, the formula (4) turns into

$$MLP\left(\mathbf{X}\right) = \sigma\left(\mathbf{X}\tilde{\mathbf{W}}_{up}\right) \cdot \tilde{\mathbf{W}}_{down}. \tag{10}$$

Ladder Omission Rate Scheduler (LORS) Empirically, pruning some neurons may pose a risk of overfitting to the model. To mitigate this potential issue, we plan to divide the training process into two stages: 1) Train sparsely with a constant omission rate and 2) Gradually decrease the omission rate, making the model denser until it fully recovers as a dense model. Through the second stage, the overfitting problem of certain parameters in the model is alleviated to some extent.

For the second stage, we experimented with linear and cosine omission rate schedulers. In terms of model performance, it is common to employ these schedulers during training. However, due to the frequent fluctuations in omission rate, there is considerable variability in training speed, occasionally resulting in slower progress compared to training without neuron omission. Therefore, we considered reducing the frequency of changes in the omission rate by maintaining a constant rate for a short period before decreasing to the next constant rate segment. Additionally, similar to curriculum learning (Bengio et al., 2009), we gradually lengthen the period to allow the model to adapt to the changes in the omission rate.

Based on these insights, we propose a Ladder Omission Rate Scheduler. Figure 3 (c) visualizes an example of LORS. Mathematically, considering a maximum neuron omission rate r and total training steps T, supposing the neuron omission rate initiates its decrease at step η with a ladder number denoted as L, the neuron omission rate r_t at step t can be formulated as follows:

$$r_t = \begin{cases} r, & t \leq \eta \\ \max\left\{0, r - \frac{r}{L} \cdot \ell_t\right\}, & t > \eta \end{cases}, \text{ where}$$
$$\frac{2^{\ell_t - 1} - 1}{2^L - 1} \leq \frac{t - \eta}{T - \eta} < \frac{2^{\ell_t} - 1}{2^L - 1} \text{ and } \ell_t \in \mathbb{N}^+. \tag{11}$$

2.3 Implementation and Theoretical Efficiency Analysis

Implementation Taking into account that we are accelerating the training of LLMs at the algorithmic level, it is imperative for SAT to be compatible with lower-level acceleration techniques, such as operator-level acceleration (e.g., flash attention (Dao et al., 2022; Dao, 2023)) and training frameworks like DeepSpeed (Rasley et al., 2020). To accomplish this, we employ an exceedingly simple and convenient implementation approach: for a linear layer, during each training iteration, the optimizer remains consistent with standard training, preserving a complete parameter matrix. Nevertheless, during computation, we exclusively engage the sparse portions (several columns or rows) of the parameter matrix for calculation.

Efficiency Analysis In theory, we compare SAT and standard training in terms of both memory consumption and FLOPs (floating point of operations).

- Memory: Given that the primary memory-intensive components on the GPU, namely the model parameters and optimizer states, remain consistent with those of standard training, the actual memory demand of our SAT during training theoretically should align closely with that of standard training.
- FLOPs: During both the forward and backward processes, we omit the computation of r proportion of neurons. Consequently, by disregarding the calculation overhead of the second stage of LORS (mentioned in subsection 2.2), SAT can save approximately $\frac{\eta r}{T}$ FLOPs.

3 Experiments

We conduct comprehensive experiments to address the following three questions:

Q1: What is the optimal configuration, including assessment of neurons importance (across various metrics) and neurons selection (top-k vs. sampling), with LORS for SAT?

Q2: Does SAT harm the performance of LLMs in both continual pre-training and supervised fine-tuning scenarios?

Q3: What is the efficiency achieved by SAT over popular LLMs in practice?

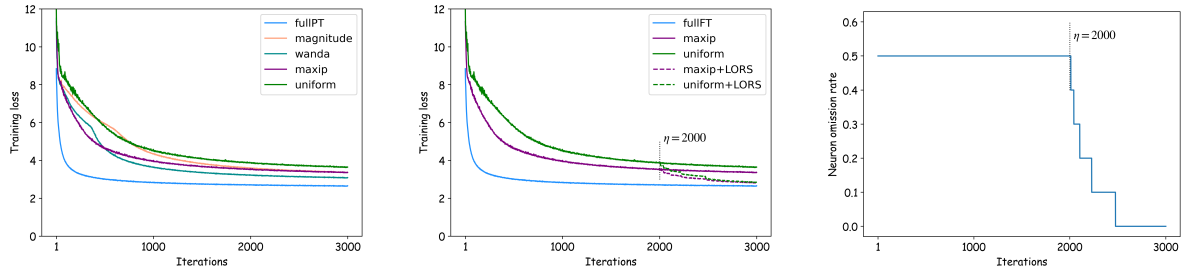
3.1 Neuron importance assessment and omission

We first conduct experiments over relatively small models to answer Q1 due to limitations in computing resources and time.

Setup We initially continue pre-training TinyLlama-1.1B (Zhang et al., 2024) over about 12B tokens of Chinese data to augment its capabilities for understanding and generating Chinese text. Inspired by Cui et al., we extend its vocabulary with an additional 33000 Chinese tokens. Following Wei et al., we validate the improvement of the Chinese language processing capability of models by evaluating perplexity over Skywork (Wei et al., 2023), which is a Chinese language modeling evaluation benchmark across 6 domains. For neuron importance assessment, we try the four aforementioned metrics: uniform, magnitude (Han et al., 2015), wanda (Sun et al., 2023) and maxip (Liu et al., 2023). For neuron selection methods, we explore both top-k and

Order	CPT	NIM	NSM	Skywork PPL ↓ finance	game	general	government	movie	technology	Avg. ↓
w/o sparsity										
1	✗	–	–	8.40	16.95	11.31	13.00	24.26	11.24	14.19
2	✓	–	–	**4.36**	**11.39**	**5.74**	**5.08**	**15.54**	**7.50**	**8.27**
w/ sparsity										
3	✓	uniform	–	6.91	18.25	8.92	8.61	25.58	11.41	13.28
4	✓	magnitude	top-k	30.92	52.59	33.26	47.71	77.75	34.61	46.14
5	✓	wanda	top-k	17.16	32.04	16.55	24.62	49.39	21.51	26.88
6	✓	maxip	top-k	20.60	56.15	23.55	39.25	78.19	31.98	41.62
7	✓	magnitude	sampling	7.68	21.19	10.06	9.42	29.20	13.27	15.14
8	✓	wanda	sampling	7.23	18.07	9.06	8.97	25.86	11.45	13.44
9	✓	maxip	sampling	**6.56**	**17.58**	**8.59**	**8.10**	**24.89**	**11.03**	**12.79**
w/ sparsity+LORS										
10	✓	uniform	–	5.69	13.51	7.20	6.72	18.79	9.31	10.20
11	✓	maxip	sampling	**4.87**	**12.54**	**6.35**	**5.82**	**17.25**	**8.19**	**9.17**

Table 1: Results of continual pre-training of TinyLlama-1.1B with different neuron importance metrics and neuron selection methods. CPT: continual pre-training; ✓and ✗denote continually pre-training the models and not, respectively. NIM: neuron importance metric. NSM: neuron selection method. LORS: ladder omission rate scheduler. The omission rate is set to 50%.



(a) Training loss under different neuron importance metrics with the sampling neuron selection method.

(b) Traning loss under uniform and maxip with LORS depicted in the right subfigure.

(c) The ladder omission rate scheduler (LORS). $T = 3000, \eta = 2000, r = 0.5$.

Figure 3: Continual pre-training of TinyLlama-1.1B

sampling. All results are obtained by the popular LM-Evaluation-Harness (Gao et al., 2023) tool.

Hyperparameters We continue pre-training TinyLlama-1.1B by Megatron-DeepSpeed (Smith et al., 2022) framework on 32 A800-80G GPUs with Zero-3 (Rajbhandari et al., 2020) and FlashAttention (Dao et al., 2022; Dao, 2023) techniques. The batch size is 1024 and the maximum length is 4096. We adopt a cosine learning rate scheduler with a maximum learning rate of 5e-5. For the ladder omission rate scheduler, $T = 3000, L = 5$, and $\eta = 2000$. For the sampling neuron selection method, we explore three different temperature settings: $[0.1, 0.05, 0.01]$. The results are shown in Table 5 of Appendix A. Finally, we set $\tau = 0.05$ in all our experiments.

Results First, from Table 1 (line 4-6 vs. 7-9), we discover the neuron selection method of sampling beats top-k under all neuron importance metrics (Q1). We suspect the top-k method tends to concentrate on selecting relatively fixed neurons which causes overfitting. Then, Figure 3 demonstrates the continual pre-training loss of TinyLlama-1.1B. From Figure 3(a), we can find the lowest losses achieved by all neuron importance metrics are higher than those of full pre-training due to the lack of half-trainable parameters in each iteration. Among all neuron importance metrics, wanda obtains the lowest training loss, while not as smooth as uniform and maxip during early stages (about 300-500 iterations). Considering the language modeling evaluation results of Table 1 (line 3, 7-9), the uniform and maxip equipped with sampling neuron selection method may be reasonable SAT configuration candidates.

Subsequently, we introduce the ladder omission rate scheduler (see Figure 3(c)). We can observe

Order	CPT	Skywork PPL ↓						Avg. ↓	CMMLU ↑	AGI-Eval ↑	Avg. ↑
		finance	game	general	government	movie	technology				
Llama-2 7B											
1	Original	6.48	12.65	7.79	8.60	18.70	8.81	10.51	32.27	30.04	31.16
2	FullCPT	3.37	8.24	4.50	3.77	11.04	5.76	6.11	32.88	31.94	32.41
3	SPDF	3.91	9.35	5.10	4.49	12.79	6.40	7.01	31.89	31.79	31.84
4	SAT (ours)	3.69	8.78	4.80	4.14	11.86	6.07	6.56	32.79	31.84	32.32
Llama-2 13B											
5	Original	6.00	11.46	7.26	7.59	17.22	8.09	9.60	38.08	37.73	37.91
6	FullCPT	3.25	7.88	4.38	3.61	10.70	5.54	5.89	44.53	40.02	42.28
7	SPDF	3.73	8.82	4.89	4.24	12.31	6.08	6.68	40.71	37.20	38.96
8	SAT (ours)	3.48	8.23	4.58	3.89	11.22	5.76	6.19	44.36	39.78	42.07

Table 2: Language modeling and Chinese benchmark evaluation results of SAT with optimal configuration in the continual pre-training scenario. CPT: continual pre-training. SAT: sparsity-accelerated training.

the training loss decreases rapidly to a level comparable to full pre-training for both configuration candidates. However, the maxip metric obtains a faster rate of loss reduction. Combining with the results in Table 1 (line 10-11), we conclude *maxip* neuron importance metric with *sampling* neuron selection method, plus the *LORS* are optimal configurations for SAT (Q1). Furthermore, SAT achieves comparable perplexity to full pre-training with such optimal configuration.

3.2 Performance effects of SAT in both scenarios

Next, we validate whether SAT with the optimal configuration works in both continual pre-training and supervised fine-tuning scenarios (Q2).

Setup To better simulate real-world application scenarios, we perform experiments on larger LLMs (7B and 13B). For the continual pre-training scenario, we adopt the identical pre-training setup to TinyLlama-1.1B. Besides the language modeling evaluation on Skywork, we evaluate LLMs on two additional popular Chinese benchmarks: CMMLU (Li et al., 2023a) and AGI-Eval (Zhong et al., 2023)[4]. For the supervised fine-tuning scenario, we instruction-tune LLMs on 50K subsample of Flan v2 (Longpre et al., 2023; Wang et al., 2023). Following Iivison et al., we assess the instruction-tuned LLMs on GPT4ALL (Anand et al., 2023) for reasoning ability and MMLU (Hendrycks et al., 2020) for factuality ability.

Hyperparameters For the continual pre-training, all pre-training hyperparameters are the same as those of TinyLlama-1.1B. We evaluate models on CMMLU and AGI-Eval in a 3-shot manner. For the supervised fine-tuning, we adopt the Huggingface+DeepSpeed (Wolf et al., 2019; Rasley et al., 2020) framework with Zero-3 technique. We instruction-tune models 3 epochs with a cosine learning rate scheduler whose maximum learning rate is 1e-5 on 64 A800-80G GPUs. The batch size is 128. Similar to PEFT methods (Hu et al., 2022; Liu et al., 2022), we only tune a small proportion of parameters at each training step. In particular, we set $r = 96\%$, $T = 1200$, $\eta = 600$, and $L = 1$. The models are evaluated in a 5-shot manner for both GPT4ALL and MMLU. More details of baselines are in Appendix B.

Results Table 2 shows the results of continual pre-training. First, the continual pre-training enhances the Chinese text-processing ability of LLMs (line 1 vs. 2; line 5 vs. 6). The models perform comparably with SAT to full pre-training (line 2 vs. 4; line 6 vs. 8). Furthermore, we adjusted to adapt SPDF (Thangarasa et al., 2023) for continual pre-training. Specifically, we conducted sparse pre-training first, followed by dense continual pre-training with the model. We can find our SAT consistently outperforms SPDF (line 3 vs. 4; line 7 vs. 8). We can state the SAT harms little to the performance of LLMs in the continual pre-training scenario (Q2).

Table 3 illustrates the results of supervised fine-tuning. The SFT boosts the performance of Llama-2 7B a lot (line 1 vs. 2) while contributing minor improvements to Llama-2 13B (line 7 vs. 8). We attribute this to the fact that Llama-2 13B, with its greater number of parameters, has a relatively strong inherent capacity before SFT (line 1, 2, and 7). Maybe it requires more SFT data to achieve further performance improvements. Compared to famous PEFT methods, $(IA)^3$, SpIEL, and LoRA, our SAT obtains better performance on GPT4ALL

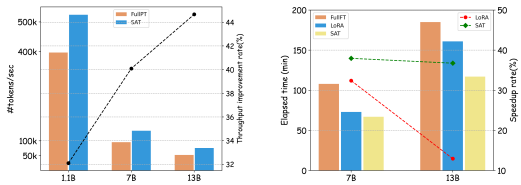
[4]We only evaluate on single-choice problems.

Order	SFT	Method	GPT4ALL Acc ↑							Avg. ↑	MMLU ↑
			HellaSwag	Obqa	WinoGrande	ARC_c	ARC_e	boolq	piqa		
Llama-2 7B											
1	✗	Original	58.30	36.40	73.80	49.23	79.29	79.36	78.78	65.02	45.8
2	✓	FullFT	**62.79**	40.00	74.82	49.32	79.17	84.01	77.80	66.84	50.5
3	✓	$(IA)^3$	58.46	36.40	74.35	48.46	79.63	80.52	79.27	65.30	46.7
4	✓	SpIEL	58.74	37.40	75.30	52.30	81.44	**84.37**	79.38	66.99	**50.7**
5	✓	LoRA	58.06	37.80	75.30	50.09	79.88	83.27	79.43	66.26	49.3
6	✓	SAT (ours)	59.71	**40.20**	**76.01**	**52.56**	**81.52**	83.85	**79.71**	**67.65**	50.5
Llama-2 13B											
7	✗	Original	61.27	37.20	77.11	52.82	82.03	83.70	79.38	67.64	55.3
8	✓	FullFT	**65.86**	40.40	74.66	52.22	80.93	83.73	79.00	68.11	55.6
9	✓	SpIEL	61.15	38.40	78.45	54.27	82.95	86.45	79.98	68.81	**55.8**
10	✓	$(IA)^3$	61.62	36.60	77.11	53.16	82.28	85.11	79.98	67.98	54.3
11	✓	LoRA	61.01	40.00	78.61	52.05	82.07	86.06	80.20	68.57	**55.8**
12	✓	SAT	62.29	**41.60**	**78.85**	**56.57**	**83.12**	**86.82**	**81.18**	**70.06**	55.4

Table 3: Factuality and Reasoning evaluation results of SAT with optimal configuration in the supervised fine-tuning scenario. SFT: supervised fine-tuning. $(IA)^3$ (Liu et al., 2022), SpIEL (Ansell et al., 2024) and LoRA (Hu et al., 2022) are popular PEFT methods for SFT. FullFT: full fine-tuning.

Scenario	Model	Saving (%)		7B			13B		
		FLOPs	FLOPS (↓)	Time (h)	Speedup (%)	PM (GB)	Time (h)	Speedup (%)	PM (GB)
CPT	FullCPT	-	-	36.5	-	44.7	66.4	-	58.4
	SAT (ours)	33	6	26.1	28.5	44.7	45.9	30.9	58.3
SFT	FullFT	-	-	1.8	-	23.9	3.1	-	36.6
	SpIEL	-	-	1.5	16.7	10.3	3.0	3.2	13.5
	LoRA	-	-	1.2	33.3	8.1	2.7	12.9	13.1
	SAT (ours)	48	16	1.1	38.9	23.1	2.0	35.5	36.4

Table 4: Efficiency of SAT on Llama-2 7B and 13B for both continual pre-training (CPT) and supervised fine-tuning (SFT). FLOPs:floating operations; FLOPS: floating operations per second; PM: Peak Memory.



(a) Throughput of continual pre-training. (b) Elapsed time of supervised fine-tuning.

Figure 4: Speedup of SAT for both continual pre-training and supervised fine-tuning.

and comparable results on MMLU. Furthermore, our SAT achieves comparable results on MMLU and even better accuracy on GPT4ALL compared to FullFT. We suspect that this may be due to the removal of some unnecessary neurons, which alleviates the risk of overfitting. Overall, the SAT has almost no bad effects on the model performance (Q2).

After confirming that SAT has little to no detrimental effect on model performance, we finally examine the actual acceleration it can achieve (Q3).

3.3 Efficiency of SAT

Setup We compare the speedup of SAT with standard training in both continual pre-training and supervised fine-tuning scenarios. All experiments are performed on identical 4 machines, each equipped with 8 A800 GPUs under the setup described in § 3.1 and § 3.2. For the continual pre-training, we compare the number of processed tokens per second since all training samples are truncated or padded to the same length. For the supervised fine-tuning, we compare the total elapsed training time due to the variable lengths of training samples.

Results Figure 4 provides the speedup results for both continual pre-training and supervised fine-tuning. For the continual pre-training, we discover SAT improves throughput increases with the growth of model sizes (the dashed line in Figure 4(a)) and the maximum throughput improvement approaches 45%. For the supervised fine-tuning, the speedup of LoRA decreases significantly when the model size is relatively large. However, our SAT is more stable. In general, SAT can obtain

about 38% speedup in elapsed time.

Table 4 presents the results in terms of theoretical FLOPs, actual FLOPS, training time, and memory usage. The FLOPS decrease is attributed to the reduced computation required for each iteration. As we can see, SAT achieved the optimal acceleration effect; however, there was no improvement in memory usage, which is consistent with the theoretical analysis and experimental results in the previous discussion. Finally, we can determine SAT indeed accelerates both continual pre-training and supervised fine-tuning (Q3).

4 Related Works

Transformer Pruning Pruning has demonstrated considerable potential in augmenting the inference speed and reducing the memory footprint of transformers, as highlighted in prior research (Kwon et al., 2022). Broadly speaking, transformer pruning techniques can be delineated into two principal categories (Chitty-Venkata et al., 2023): unstructured pruning and structured pruning. Unstructured pruning methods (Gordon et al., 2020; Campos et al., 2022; Zhang et al., 2022a) possess the capability to substantially diminish the number of parameters due to their fine-grained pruning granularity. Nonetheless, they frequently encounter challenges in achieving significant improvements in inference speed across diverse hardware platforms (Sanh et al., 2020).

Hence, numerous researchers are embracing structured pruning methods at the granularity level of entire layers, filters, channels, or heads (Li et al., 2021). For instance, Liu et al. prunes the unimportant heads within multi-head attention layers and insignificant channels in feed-forward networks on BERT (Kenton and Toutanova, 2019). Liu et al. and Song et al. extend such a method and scale it for large language models. In this study, distinct from their focus on the inference of large language models (LLMs), we draw upon the principles of structured pruning and apply them to the training of LLMs.

Sparse Fine-Tuning Sparse fine-tuning is a technique aimed at expediting model fine-tuning and diminishing model memory demands by selectively updating only a small subset of parameters (Ansell et al., 2022; Li et al., 2024). For relatively small language models, such as BERT, Guo et al. devise a binary mask controlled by a regularization term to manage the parameters for updating. Sung et al. induce the mask by retaining parameters exhibiting higher Fisher information over numerous iterations and subsequently maintain the mask to sparsely fine-tune models. Additionally, Ansell et al. retain parameters that undergo the most significant changes during an initial round of full fine-tuning.

With the widespread adoption of LLMs, Ansell et al. and Zhao et al. commence scaling sparse fine-tuning to LLMs. However, to mitigate the memory overhead of LLMs, they both incorporate the parameter-efficient fine-tuning (PEFT) methods (Houlsby et al., 2019; Li and Liang, 2021; Lialin et al., 2023), which may not always be entirely suitable (Cui et al., 2023; Zhao et al., 2024b). In this work, instead of employing the PEFT methods, we place greater emphasis on conventional sparse training techniques, making it applicable to common scenarios such as continual pre-training and supervised fine-tuning for LLMs.

5 Conclusion

In this paper, we aim to accelerate two regular additional training scenarios of LLMs, i.e., continual pre-training and supervised fine-tuning. Our main intuition is to leverage the sparsity of neurons in large models, as discovered by previous researchers, to accelerate training by disregarding computations for unimportant neurons and propose a sparsity-accelerated training (SAT) framework. Extensive experiments on several benchmarks demonstrate SAT can accelerate the training of LLMs and maintain performance simultaneously.

Limitations

On one hand, this work primarily focuses on large language models and can explore even larger and more diverse model structures in the future. Furthermore, several other neuron importance metrics are necessary to explore. On the other hand, training numerous large-scale models requires substantial computational resources, incurring high costs and resulting in significant carbon emissions.

Acknowledgements

We thank all the reviewers for their valuable suggestions on this work. This work is funded by the China NSFC Projects (92370206, U23B2057,62106142 and 62120106006) and Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102).

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Yuvanesh Anand, Zach Nussbaum, Adam Treat, Aaron Miller, Richard Guo, Ben Schmidt, GPT4All Community, Brandon Duderstadt, and Andriy Mulyar. 2023. Gpt4all: An ecosystem of open source compressed language models.

Alan Ansell, Edoardo Ponti, Anna Korhonen, and Ivan Vulić. 2022. Composable sparse fine-tuning for cross-lingual transfer. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1778–1796, Dublin, Ireland. Association for Computational Linguistics.

Alan Ansell, Ivan Vulić, Hannah Sterz, Anna Korhonen, and Edoardo M Ponti. 2024. Scaling sparse fine-tuning to large language models. *arXiv preprint arXiv:2401.16405*.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

Daniel Campos, Alexandre Marques, Tuan Nguyen, Mark Kurtz, and ChengXiang Zhai. 2022. Sparse* bert: Sparse models are robust. *arXiv preprint arXiv:2205.12452*.

Krishna Teja Chitty-Venkata, Sparsh Mittal, Murali Emani, Venkatram Vishwanath, and Arun K Somani. 2023. A survey of techniques for optimizing transformer inference. *Journal of Systems Architecture*, page 102990.

Yiming Cui, Ziqing Yang, and Xin Yao. 2023. Efficient and effective text encoding for chinese llama and alpaca. *arXiv preprint arXiv:2304.08177*.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359.

Marc Dillmann, Julien Siebert, and Adam Trendowicz. 2024. Evaluation of large language models for assessing code maintainability. *arXiv preprint arXiv:2401.12714*.

Guanting Dong, Hongyi Yuan, Keming Lu, Chengpeng Li, Mingfeng Xue, Dayiheng Liu, Wei Wang, Zheng Yuan, Chang Zhou, and Jingren Zhou. 2023. How abilities in large language models are affected by supervised fine-tuning data composition. *arXiv preprint arXiv:2310.05492*.

Robin Dupont, Mohammed Amine Alaoui, Hichem Sahbi, and Alice Lebois. 2022. Extracting effective subnetworks with gumbel-softmax. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 931–935. IEEE.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

Mitchell A Gordon, Kevin Duh, and Nicholas Andrews. 2020. Compressing bert: Studying the effects of weight pruning on transfer learning. *arXiv preprint arXiv:2002.08307*.

Demi Guo, Alexander Rush, and Yoon Kim. 2021. Parameter-efficient transfer learning with diff pruning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4884–4896, Online. Association for Computational Linguistics.

Kshitij Gupta, Benjamin Thérien, Adam Ibrahim, Mats L Richter, Quentin Anthony, Eugene Belilovsky, Irina Rish, and Timothée Lesort. 2023. Continual pretraining of large language models: How to (re) warm your model? *arXiv preprint arXiv:2308.04014*.

Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi,

Joel Jang, David Wadden, Noah A Smith, Iz Beltagy, et al. 2023. Camels in a changing climate: Enhancing lm adaptation with tulu 2. *arXiv preprint arXiv:2311.10702*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b.

Zixuan Ke, Yijia Shao, Haowei Lin, Tatsuya Konishi, Gyuhak Kim, and Bing Liu. 2023. Continual pre-training of language models. In *The Eleventh International Conference on Learning Representations*.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, page 2.

Woosuk Kwon, Sehoon Kim, Michael W Mahoney, Joseph Hassoun, Kurt Keutzer, and Amir Gholami. 2022. A fast post-training pruning framework for transformers. *Advances in Neural Information Processing Systems*, 35:24101–24116.

Hanqi Li, Lu Chen, Da Ma, Zijian Wu, Su Zhu, and Kai Yu. 2024. Evolving subnetwork training for large language models. In *Forty-first International Conference on Machine Learning*.

Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. 2023a. Cmmlu: Measuring massive multitask language understanding in chinese. *arXiv preprint arXiv:2306.09212*.

Jiaoda Li, Ryan Cotterell, and Mrinmaya Sachan. 2021. Differentiable subset pruning of transformer heads. *Transactions of the Association for Computational Linguistics*, 9:1442–1459.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Zonglin Li, Chong You, Srinadh Bhojanapalli, Daliang Li, Ankit Singh Rawat, Sashank Reddi, Ke Ye, Felix Chern, Felix Yu, Ruiqi Guo, and Sanjiv Kumar. 2023b. The lazy neuron phenomenon: On emergence of activation sparsity in transformers. In *Conference on Parsimony and Learning (Recent Spotlight Track)*.

Vladislav Lialin, Vijeta Deshpande, and Anna Rumshisky. 2023. Scaling down to scale up: A guide to parameter-efficient fine-tuning.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965.

Zejian Liu, Fanrong Li, Gang Li, and Jian Cheng. 2021. Ebert: Efficient bert inference with dynamic structured pruning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4814–4823.

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

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, and Adam Roberts. 2023. The flan collection: designing data and methods for effective instruction tuning. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Rui Mao, Guanyi Chen, Xulang Zhang, Frank Guerin, and Erik Cambria. 2023. Gpteval: A survey on assessments of chatgpt and gpt-4. *arXiv preprint arXiv:2308.12488*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506.

Victor Sanh, Thomas Wolf, and Alexander Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning. *Advances in Neural Information Processing Systems*, 33:20378–20389.

Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, et al. 2022. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990*.

Yixin Song, Zeyu Mi, Haotong Xie, and Haibo Chen. 2023. Powerinfer: Fast large language model serving with a consumer-grade gpu. *arXiv preprint arXiv:2312.12456*.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2023. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*.

Yi-Lin Sung, Varun Nair, and Colin A Raffel. 2021. Training neural networks with fixed sparse masks. *Advances in Neural Information Processing Systems*, 34:24193–24205.

Vithursan Thangarasa, Abhay Gupta, William Marshall, Tianda Li, Kevin Leong, Dennis DeCoste, Sean Lie, and Shreyas Saxena. 2023. Spdf: Sparse pre-training and dense fine-tuning for large language models. In *Uncertainty in Artificial Intelligence*, pages 2134–2146. PMLR.

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

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

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Raghavi Chandu, David Wadden, Kelsey MacMillan, Noah A Smith, Iz Beltagy, et al. 2023. How far can camels go? exploring the state of instruction tuning on open resources. *arXiv preprint arXiv:2306.04751*.

Tianwen Wei, Liang Zhao, Lichang Zhang, Bo Zhu, Lijie Wang, Haihua Yang, Biye Li, Cheng Cheng, Weiwei Lü, Rui Hu, Chenxia Li, Liu Yang, Xilin Luo, Xuejie Wu, Lunan Liu, Wenjun Cheng, Peng Cheng, Jianhao Zhang, Xiaoyu Zhang, Lei Lin, Xiaokun Wang, Yutuan Ma, Chuanhai Dong, Yanqi Sun, Yifu Chen, Yongyi Peng, Xiaojuan Liang, Shuicheng Yan, Han Fang, and Yahui Zhou. 2023. Skywork: A more open bilingual foundation model.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Chaoyi Wu, Weixiong Lin, Xiaoman Zhang, Ya Zhang, Yanfeng Wang, and Weidi Xie. 2023. Pmc-llama: Towards building open-source language models for medicine. *arXiv preprint arXiv:2305.10415*, 6.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*.

Qingru Zhang, Simiao Zuo, Chen Liang, Alexander Bukharin, Pengcheng He, Weizhu Chen, and Tuo Zhao. 2022a. Platon: Pruning large transformer models with upper confidence bound of weight importance. In *International Conference on Machine Learning*, pages 26809–26823. PMLR.

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

Bowen Zhao, Hannaneh Hajishirzi, and Qingqing Cao. 2024a. Apt: Adaptive pruning and tuning pretrained language models for efficient training and inference. *arXiv preprint arXiv:2401.12200*.

Zihan Zhao, Da Ma, Lu Chen, Liangtai Sun, Zihao Li, Hongshen Xu, Zichen Zhu, Su Zhu, Shuai Fan, Guodong Shen, Xin Chen, and Kai Yu. 2024b. Chemdfm: Dialogue foundation model for chemistry.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2023. Agieval: A human-centric benchmark for evaluating foundation models. *arXiv preprint arXiv:2304.06364*.

A Temperature Exploration

Temperature	Skywork PPL ↓						Avg. ↓
	finance	game	general	government	movie	technology	
0.1	6.71	18.05	8.74	8.32	25.68	11.24	13.12
0.05	**6.56**	**17.58**	**8.59**	**8.10**	**24.89**	**11.03**	**12.79**
0.01	9.87	23.16	11.51	14.06	34.90	14.74	18.04

Table 5: Perplexity scores on the Skywork benchmark of TinyLLaMA (1.1B) after continual pre-training with different temperature settings with the maxip neuron importance metric

B Baselines Hyperparameters in Supervised Fine-tuning

Model	7B			13B		
	lr	r	λ	lr	r	λ
$(IA)^3$	3e-4	-	-	1e-4	-	-
LoRA	1e-4	64	-	3e-5	64	-
SpIEL	1e-4	64	30	1e-5	64	30
FullFT	1e-5	-	-	1e-5	-	-
SAT (ours)	1e-5	-	-	1e-5	-	-

Table 6: Baselines hyperparameters in supervised fine-tuning for LLaMA-2 7B and 13B. lr: learning rate; r: rank in LoRA; λ: weight decay strength in SpIEL