arXiv:2505.24037v1 [cs.AI] 29 May 2025

Leave it to the Specialist: Repair Sparse LLMs with Sparse Fine-Tuning via Sparsity Evolution

Qiao Xiao[1], **Alan Ansell**[2], **Boqian Wu**[3,4], **Lu Yin**[5,1], **Mykola Pechenizkiy**[1], **Shiwei Liu**[6,1], **Decebal Constantin Mocanu**[3,1]
[1]Eindhoven University of Technology, [2]University of Cambridge,
[3]University of Luxembourg, [4]University of Twente,
[5]University of Surrey, [6]University of Oxford
{q.xiao,m.pechenizkiy}@tue.nl, aja63@cam.ac.uk,
b.wu@utwente.nl, shiwei.liu@maths.ox.ac.uk,
l.yin@surrey.ac.uk, decebal.mocanu@uni.lu

Abstract

Large language models (LLMs) have achieved remarkable success across various tasks but face deployment challenges due to their massive computational demands. While post-training pruning methods like SparseGPT and Wanda can effectively reduce the model size, but struggle to maintain model performance at high sparsity levels, limiting their utility for downstream tasks. Existing fine-tuning methods, such as full fine-tuning and LoRA, fail to preserve sparsity as they require updating the whole dense metrics, not well-suited for sparse LLMs. In this paper, we propose **Sparsity Evolution Fine-Tuning (SEFT)**, a novel method designed specifically for sparse LLMs. SEFT dynamically evolves the sparse topology of pruned models during fine-tuning, while preserving the overall sparsity throughout the process. The strengths of SEFT lie in its ability to perform task-specific adaptation through a weight drop-and-grow strategy, enabling the pruned model to self-adapt its sparse connectivity pattern based on the target dataset. Furthermore, a sensitivity-driven pruning criterion is employed to ensure that the desired sparsity level is consistently maintained throughout fine-tuning. Our experiments on various LLMs, including LLaMA families, DeepSeek, and Mistral, across a diverse set of benchmarks demonstrate that SEFT achieves stronger performance while offering superior memory and time efficiency compared to existing baselines. Our code is publicly available at: https://github.com/QiaoXiao7282/SEFT.

1 Introduction

Large language models (LLMs) have recently demonstrated remarkable success in a wide range of complex tasks, showcasing capabilities that span language understanding [Zhu et al., 2023], reasoning [Wei et al., 2022], and code generation [Vaithilingam et al., 2022], etc. However, these models, characterized by their immense number of parameters, pose significant challenges for real-world deployment due to their computational and memory demands.

To address these challenges, researchers have proposed post-training pruning approaches to reduce the number of parameters in large language models (LLMs) without incurring the high costs of retraining [Ma et al., 2023, Naveed et al., 2023, Zhu et al., 2024]. For example, SparseGPT, as outlined in [Frantar and Alistarh, 2023], minimizes the discrepancy between pruned and original dense models using a layer-wise Hessian method. Furthermore, Wanda [Sun et al., 2023] introduces a metric combining output activations with weight magnitudes to identify unimportant parameters. However, such methods often fail to maintain the performance of pre-trained models, especially under high

Preprint. Under review.

sparsity levels (e.g., $\geq 60\%$) [Yin et al., 2023, Lu et al., 2024a]. This performance degradation poses a significant challenge to their practical application, despite the fact that pruning can substantially improve model efficiency.

While parameter-efficient fine-tuning (PEFT) has been widely adopted to fine-tune LLMs in a computationally efficient manner [Mangrulkar et al., 2022, Pfeiffer et al., 2023, Ding et al., 2023, Han et al., 2024], common approaches like LoRA [Hu et al., 2021] and its variants [Xu et al., 2023, Zhao et al., 2024, Dettmers et al., 2024] often fall short of full fine-tuning in large-scale settings [Biderman et al., 2024]. Moreover, when directly applied to fine-tune sparse LLMs, these methods either incur additional storage overhead to retain the fine-tuned parameters or result in dense models after merging—which conflicts with the sparse model structure and undermines the efficiency benefits of maintaining sparsity, as illustrated in Figure 1 (a).

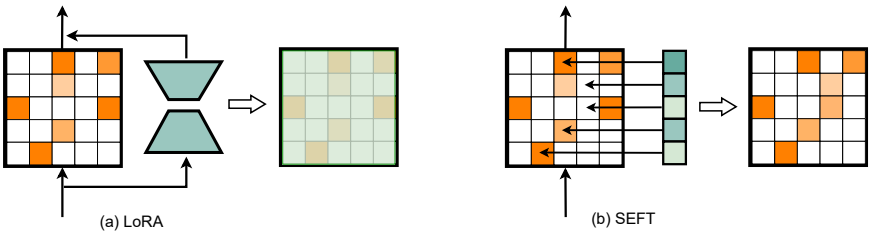

Figure 1: Comparison of LoRA and SEFT fine-tuning for sparse LLMs. While LoRA-based methods require low-rank matrices, SEFT utilizes indices (arrows) and corresponding deltas (green squares) to update LLM parameters.

More recently, Sparsity-Preserved Parameter-Efficient Fine-Tuning methods [Lu et al., 2024c, Munoz et al., 2024, Hu et al., 2025] have been proposed to maintain sparsity after fine-tuning by incorporating specially designed adaptation mechanisms. However, these methods typically assume that the sparse topology produced by post-training pruning remains consistent across the training of downstream tasks. Given the heavy reliance on calibration data in state-of-the-art LLM pruning methods, the pruning process may inadvertently remove connections that are crucial for specific downstream tasks, leading to suboptimal performance.

This raises a fundamental question: *Can we efficiently fine-tune sparse LLMs while dynamically evolving their sparse topology to better adapt to downstream tasks?*

Inspired by Dynamic Sparse Training (DST) [Mocanu et al., 2018, Evci et al., 2020], which evolves sparse connectivity efficiently during training, we propose Sparsity Evolution Fine-Tuning (SEFT). SEFT extends the principles of DST to the fine-tuning of sparse LLMs by enabling dynamic sparsity evolution. This allows the sparse LLMs to recover task-relevant connections that may have been pruned and to adapt its sparse topology throughout fine-tuning, as illustrated in Figure 1 (b).

Unlike LoRA-based approaches, SEFT directly modifies LLM parameters using a set of learnable indices and corresponding updates. During fine-tuning, SEFT operates in two distinct phases: sparse topology evolution and sparsity adaptation. In the sparse topology evolution phase, parameters with smaller updates, determined by changes in magnitude, are dropped, while new connections are grown based on gradient information. Notably, previously inactive weights in the sparse model are allowed to be reactivated during this growth step, enabling sparse topology evolution to the target task. Following this, the sparsity adaptation phase restores the model to the desired sparsity level using a sensitivity-based pruning criterion. This step ensures that the model remains within the predefined sparsity budget while preserving performance.

We evaluate SEFT on a variety of sparse LLMs from the LLaMA [Touvron et al., 2023a,b, Dubey et al., 2024] families, as well as DeepSeek [Bi et al., 2024] and Mistral [Jiang et al., 2023], with sparsity introduced via two state-of-the-art post-training pruning methods: SparseGPT and Wanda. Our experiments span models of different sizes, including 7B, 8B, and 13B, and demonstrate the effectiveness of SEFT across a diverse set of fine-tuning benchmarks, including commonsense reasoning tasks, MMLU, and GSM8K. Our contributions in this work can be summarized as follows:

- We propose SEFT, an efficient fine-tuning approach tailored to sparse LLMs, which dynamically evolves the sparse topology to enable task-specific adaptation and improve downstream performance.

- We introduce a sparsity adaptation mechanism based on a sensitivity-guided pruning criterion, which preserves the target sparsity level without sacrificing performance.
- We demonstrate substantial improvements in memory and training efficiency, showing that SEFT significantly reduces resource consumption compared to prior sparsity-preserving fine-tuning methods.
- We conduct comprehensive experiments across multiple sparse LLM architectures (LLaMA, DeepSeek, and Mistral) and tasks (commonsense reasoning, MMLU, and GSM8K), validating SEFT's generality and effectiveness in both general-purpose and task-specific settings.

2 Background

2.1 Parameter-Efficient Fine-Tuning (PEFT)

Parameter-efficient fine-tuning (PEFT) has emerged as a popular approach to adapt LLMs to downstream tasks while significantly reducing the computational and memory overhead associated with traditional dense full fine-tuning [Pfeiffer et al., 2023, Ding et al., 2023, Han et al., 2024]. Instead of updating all parameters of a pre-trained model, PEFT methods introduce additional trainable parameters, which typically are a small fraction of the original model, while keeping the base model weights frozen. This significantly reduces computational overhead during fine-tuning.

One widely used PEFT method is Low-Rank Adaptation (LoRA) [Hu et al., 2021]. In LoRA, low-rank matrices are introduced into specific layers (e.g., attention or feedforward layers) to capture task-specific adaptations. Specifically, let $\mathbf{W} \in \mathbb{R}^{d\times d}$ represent a frozen weight matrix in a pre-trained model. LoRA introduces a low-rank decomposition:

$$\mathbf{W}' = \mathbf{W} + \Delta\mathbf{W}, \quad \Delta\mathbf{W} = \mathbf{A}\mathbf{B} \tag{1}$$

where $\mathbf{A} \in \mathbb{R}^{d\times r}$ and $\mathbf{B} \in \mathbb{R}^{r\times d}$ are low-rank matrices ($r \ll d$), and only $\mathbf{A}$ and $\mathbf{B}$ are trainable.

While PEFT methods like LoRA have proven effective for dense models, they are inherently not well-suited for sparse LLMs, as they either require dense updates or fail to preserve sparsity after fine-tuning.

2.2 Sparse Fine-Tuning (SFT)

Unlike LoRA-based approaches, Sparse Fine-Tuning (SFT) [Guo et al., 2021, Xu et al., 2021, Ansell et al., 2022] directly modifies the base model by introducing task-specific updates in the form of a sparse "delta" vector $\boldsymbol{\delta} \in \mathbb{R}^{d_\theta}$, which is added to the pre-trained parameters $\boldsymbol{\theta} \in \mathbb{R}^{d_\theta}$. The resulting fine-tuned model can be expressed as:

$$f'(\cdot;\boldsymbol{\theta}) = f(\cdot;\boldsymbol{\theta}+\boldsymbol{\delta}), \tag{2}$$

Here, $\boldsymbol{\delta}$ contains non-zero updates only at specific locations in $\boldsymbol{\theta}$, allowing for a sparse parameterization that enables efficient fine-tuning in practice.

Sparse Parameterization. A sparse delta vector $\boldsymbol{\delta}$ with d_ϕ non-zero entries can be efficiently represented using two components:

- Indices of non-zero values ($\boldsymbol{\eta}$): A vector of unique indices $\boldsymbol{\eta} \in \{1, 2, \ldots, d_\theta\}^{d_\phi}$, representing the positions within $\boldsymbol{\theta}$ where updates are applied.
- Update values ($\boldsymbol{\phi}$): The corresponding update values at these positions, denoted as $\boldsymbol{\phi} \in \mathbb{R}^{d_\phi}$.

Here, d_ϕ is typically a user-defined hyperparameter that controls the number of parameters updated during fine-tuning. Since $\boldsymbol{\delta}$ modifies only a small subset of the full parameter vector $\boldsymbol{\theta}$, we have $d_\phi \ll d_\theta$. This sparsity ensures that the fine-tuning process is computationally efficient and lightweight.

Optimization Objective. Sparse fine-tuning can be formulated as an optimization problem that aims to identify the optimal indices $\boldsymbol{\eta}$ and corresponding update values $\boldsymbol{\phi}$ to improve model performance on a given task-specific dataset $\mathcal{D}$. The objective is defined as:

$$\boldsymbol{\eta}^\star, \boldsymbol{\phi}^\star = \arg\min_{\boldsymbol{\eta},\boldsymbol{\phi}} \mathcal{L}(\mathcal{D};\boldsymbol{\theta},\boldsymbol{\eta},\boldsymbol{\phi}), \tag{3}$$

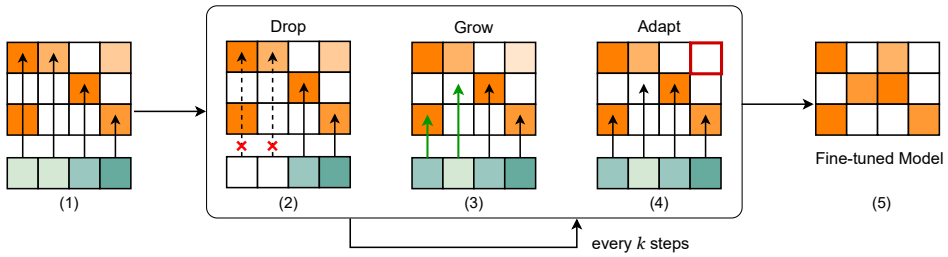

Figure 2: An illustration of SEFT for sparse LLMs. The SEFT process consists of: (1) starting with a sparse LLM, along with learnable indices (arrows) and corresponding deltas (green squares) applied to the LLM parameters; (2) dropping obsolete indices (dashed arrows); (3) growing new indices (green arrows); (4) adapting parameters to the target sparsity level (red squares). These steps are repeated every k steps. The final output is a sparse LLM after fine-tuning.

where $\mathcal{L}$ denotes the loss function that measures the discrepancy between model predictions and ground truth labels. This formulation jointly determines *which* parameters to update (through $\boldsymbol{\eta}$) and *how* to update them (through $\boldsymbol{\phi}$), while maintaining a limited number of updates for efficiency. Recent studies have scaled and validated this sparse optimization framework in the context of dense LLMs [Ansell et al., 2024], paving the way for broader exploration in sparsity-aware scenarios.

3 Sparsity Evolution Fine-Tuning for Sparse LLMs

Building upon the paradigm of sparse fine-tuning, we propose Sparsity Evolution Fine-Tuning (SEFT), a novel approach that enhances the adaptation of sparse large language models (LLMs) by allowing the dynamic evolution of their sparse topologies during task-specific fine-tuning. Given pre-trained model weights $\boldsymbol{\theta} \in \mathbb{R}^{d_\theta}$, sparsified through post-training pruning, the resulting sparse model is represented as $\boldsymbol{\theta}' = \boldsymbol{\theta} \odot \boldsymbol{M_0}$, where $\boldsymbol{M_0} \in 0, 1^{d_\theta}$ is a binary mask indicating active ($M_0 = 1$) and inactive ($M_0 = 0$) parameters. The sparsity level ρ is defined as the proportion of pruned (inactive) parameters relative to the total number of parameters in the model. We reformulate sparse fine-tuning for such models as:

$$f'(\cdot; \boldsymbol{\theta}') = f(\cdot; \boldsymbol{\theta}' + \boldsymbol{\delta}), \tag{4}$$

where $\boldsymbol{\delta}$ represents the learnable delta that introduces task-specific updates to the sparse model. Note that, SEFT explicitly allows $\boldsymbol{\delta}$ to modify both active ($M_0 = 1$) and previously pruned ($M_0 = 0$) weights, enabling the model to recover useful parameters and adapt its sparse structure during fine-tuning. To support this, SEFT introduces two complementary components: *Sparse Topology Evolution*, which explores and activates task-relevant weights, and *Sparsity Adaptation*, which prunes less important weights to restore the target sparsity level.

3.1 Sparse Topology Evolution

Inspired by the principles of Dynamic Sparse Training (DST) [Mocanu et al., 2018, Evci et al., 2020], SEFT periodically updates the indices of non-zero elements in $\boldsymbol{\delta}$ through a drop-and-grow mechanism to dynamically evolve the sparse topology during sparse LLM fine-tuning.

Drop. In each update cycle, weights corresponding to the smallest delta magnitudes are dropped from the active set:

$$\boldsymbol{\eta}_{\text{drop}} = \text{argtopk}_{\tau(t)} \left(-|\phi_i|\right), \tag{5}$$

where $|\phi_i|$ denotes the absolute value of the i-th element in the delta vector, and $\tau(t)$ determines the number of indices to drop at step t. The intuition is that smaller delta values reflect minimal task-driven adjustments to the pre-trained model, indicating that these weights contribute less to the adaptation process and can be safely pruned.

Grow. Simultaneously, an equal number of new weights $\tau(t)$, are added to the active set by selecting those with the largest gradient magnitudes:

$$\boldsymbol{\eta}_{\text{grow}} = \text{argtopk}_{\tau(t)} \left|\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{x}; \boldsymbol{\theta}' + \boldsymbol{\delta})\right|, \tag{6}$$

where $\mathcal{L}$ is the task-specific loss evaluated on the current training batch $\boldsymbol{x}$.

This approach enables the delta vector $\boldsymbol{\delta}$ to dynamically explore and activate updates for both active ($M_0 = 1$) and previously pruned ($M_0 = 0$) weights. By evolving the sparse topology during fine-tuning, SEFT enhances the model's adaptability, allowing it to recover task-relevant connections and better align its parameter allocation with downstream objectives.

3.2 Sparsity Adaptation

While the sparse topology evolution process enables the model to dynamically adapt to downstream tasks, it can lead to a denser model by activating previously inactive weights. To address this, we employ a sparsity adaptation process to restore the model to the desired sparsity level, ensuring it adheres to the predefined sparsity constraints.

In our method, we adopt a sensitivity-based pruning approach. Sensitivity, as outlined in prior research [Lee et al., 2019, Wu et al., 2023, Nowak et al., 2024], measures the importance of a parameter by evaluating its impact on the model's output or loss function when set to zero. Parameters with low sensitivity are considered redundant and can be pruned without substantially affecting the overall performance.

Algorithm 1 Sparsity Evolution Fine-Tuning (SEFT)

Input: Sparse pre-trained model weights $\boldsymbol{\theta}'$ with initial mask $\boldsymbol{M_0}$, target sparsity level ρ, delta vector $\boldsymbol{\delta}$ initialized as zeros, number of training steps T, update frequency k.
Output: Fine-tuned sparse model $\boldsymbol{\theta}'_T + \boldsymbol{\delta}$.
for $t = 1$ to T **do**
  Update $\boldsymbol{\delta}$ using task-specific loss $\mathcal{L}$.
  if $t \bmod k = 0$ **then**
    Sparse Topology Evolution
      Drop indices $\boldsymbol{\eta}_{\text{drop}}$ based on Eq. 5.
      Grow indices $\boldsymbol{\eta}_{\text{grow}}$ based on Eq. 6.
      Update $\boldsymbol{\delta}$ with $\boldsymbol{\eta}_{\text{drop}}$ and $\boldsymbol{\eta}_{\text{grow}}$.
    Sparsity Adaptation
      Calculate sensitivity for each weight as in Eq. 7.
      Get new mask $\boldsymbol{M_t}$ followed Eq. 8.
      Update sparse weights: $\boldsymbol{\theta}'_t = \boldsymbol{\theta} \odot \boldsymbol{M_t}$.
  end if
end for
Return: Fine-tuned sparse model $\boldsymbol{\theta}'_T + \boldsymbol{\delta}$.

Sensitivity Calculation: For each weight w_i, its sensitivity is computed as:

$$s_i = \left|\nabla_{\theta_i}\mathcal{L} \cdot \theta_i\right|, \tag{7}$$

where $\nabla_{\theta_i}\mathcal{L}$ is the gradient of the task-specific loss with respect to weight θ_i. This metric captures both the magnitude of the parameter and its contribution to the loss.

Pruning Decision: Weights with the smallest sensitivity values are pruned to meet the target sparsity level ρ. The new mask $\boldsymbol{M_t}$ is updated as:

$$M_i = \begin{cases} 0, & \text{if } s_i \text{ is among the smallest } \rho \text{ fraction of all } s, \\ 1, & \text{otherwise.} \end{cases} \tag{8}$$

The resulting sparse model is then represented as $\boldsymbol{\theta}'_t = \boldsymbol{\theta} \odot \boldsymbol{M_t}$.

This two-stage mechanism, sparse topology evolution followed by sparsity adaptation, enables SEFT to maintain a task-adaptive sparse structure while strictly adhering to the sparsity budget. In practice, SEFT employs accumulated gradient estimates (as [Ansell et al., 2024]) instead of instantaneous gradients, which aims to improve training stability during fine-tuning. An overview of SEFT is provided in Algorithm 1.

4 Experiments and Results

Sparse LLMs. We evaluate on several widely-used open-source LLMs, including the LLaMA family, DeepSeek, and Mistral. These models are pruned using state-of-the-art post-training pruning methods, such as SparseGPT and Wanda, to achieve the desired sparsity levels. Our focus is primarily on unstructured sparsity as it allows for fine-grained control over the parameters and has shown strong empirical results in prior works [Frantar and Alistarh, 2023, Sun et al., 2023, Yin et al., 2023]. We target highly sparse models (e.g., sparsity ≥ 0.6), where performance degradation becomes more pronounced, necessitating fine-tuning. Moreover, high sparsity levels are generally more hardware-friendly for practical speedup [Gale et al., 2020], as demonstrated in Section 4.1.3.

Fine-tuning Tasks. To comprehensively evaluate the effectiveness of SEFT, we conduct experiments across two categories of fine-tuning tasks:

(1) Performance recovery: This setting assesses the ability of fine-tuning methods to recover performance lost due to pruning. Specifically, models are fine-tuned on a general pretraining dataset, using 30k randomly selected samples from the C4 corpus [Raffel et al., 2020], and evaluated using Wikitext perplexity (PPL) [Merity et al., 2016] and Commonsense Reasoning benchmark [Gao et al., 2021].

(2) Supervised fine-tuning: This setting evaluates the models' ability to learn task-specific information through supervised data: (i) Commonsense Reasoning: Models are fine-tuned on a 170k-sample commonsense reasoning dataset and evaluated on zero-shot performance across seven commonsense reasoning tasks: BoolQ, RTE, HellaSwag, WinoGrande, ARC-e, ARC-c, and OBQA, following the evaluation protocol in [Hu et al., 2023]. (ii) MMLU: The Massive Multitask Language Understanding benchmark [Hendrycks et al., 2020] includes 57 tasks spanning diverse domains, such as elementary mathematics, US history, computer science, and law. We adopt a zero-shot evaluation protocol, where models are fine-tuned on an auxiliary training dataset and evaluated on MMLU. (iii) GSM8K [Cobbe et al., 2021]: This benchmark consists of grade-school-level math word problems. Models are fine-tuned on the training split of GSM8K and evaluated in a zero-shot setting.

Baselines. To evaluate the effectiveness of SEFT, we first obtain sparse LLMs and subsequently fine-tune them using the following sparsity-preserving baselines:

(1) LoRA*: After LoRA fine-tuning, we apply Wanda pruning post-hoc to restore the desired sparsity level and retain computational efficiency.

(2) SPP: This method fine-tunes sparse models using designed sparsity-preserving adapter, allowing the final model to maintain the target sparsity level after merging [Lu et al., 2024c].

(3) SQFT: We use a sparsity-preserving version from [Munoz et al., 2024] that excludes quantization of the base model. It maintains the target sparsity during fine-tuning by applying a binary mask.

For fair comparison, all fine-tuned LLMs are restored to target sparsity level prior to evaluation, and all fine-tuning methods are configured with the same number of trainable parameters, matched to that of LoRA under the corresponding rank configuration. We adopt a fixed rank of 32 as the default setting. Additional experimental details are provided in Appendix H.

4.1 Overall Performance

In this section, we present a comprehensive evaluation of SEFT across a range of fine-tuning scenarios. We begin by analyzing its ability to recover performance after pruning, followed by results on supervised fine-tuning tasks. Finally, we assess its efficiency in terms of memory cost, training speed, and inference latency.

4.1.1 Performance recovery

We validate the effectiveness of SEFT for performance recovery by fine-tuning sparse LLMs on a pretraining dataset, using 30k randomly sampled examples from C4 [Raffel et al., 2020]. To obtain sparse models, we first apply post-training pruning methods including Wanda and SparseGPT to reach a 70% sparsity level.

We use two evaluation metrics: LM-eval, which reports zero-shot accuracy across seven tasks from the EleutherAI LM Harness [Gao et al., 2021], and PPL, which denotes the perplexity on Wikitext-2 [Merity et al., 2016]. As shown in Table 1, fine-tuning significantly improves the performance of sparse LLMs, as indicated by lower perplexity and higher LM-eval scores compared to their unfinetuned counterparts. Notably, SEFT demonstrates strong performance recovery across various settings. It

Table 1: Performance comparison of different methods with and without fine-tuning applied to sparse LLaMA models at a sparsity level of $\rho = 0.7$.

LLaMA	Method	PPL (↓)	LM-eval (↑)
V2-7B	Wanda	75.19	34.83
	Wanda+LoRA*	11.71	44.29
	Wanda+SPP	11.53	44.46
	Wanda+SQFT	11.94	44.73
	Wanda+SEFT	**11.19**	**45.61**
	SparseGPT	27.31	41.51
	SparseGPT+LoRA*	12.86	45.42
	SparseGPT+SPP	11.58	47.27
	SparseGPT+SQFT	11.71	47.09
	SparseGPT+SEFT	**11.00**	**47.95**
V3-8B	Wanda	120.20	34.92
	Wanda+LoRA*	18.28	43.32
	Wanda+SPP	16.73	43.68
	Wanda+SQFT	17.16	43.56
	Wanda+SEFT	**16.17**	**44.55**
	SparseGPT	43.25	41.70
	SparseGPT+LoRA*	17.81	46.11
	SparseGPT+SPP	15.25	48.33
	SparseGPT+SQFT	16.14	47.70
	SparseGPT+SEFT	**15.09**	**48.89**

Table 2: Comparison of fine-tuning methods on sparse LLaMA models at sparsity level $\rho = 0.6$. Results report average and per-task zero-shot accuracy on seven commonsense reasoning tasks. Higher values indicate better performance.

LLaMA	Method	WG	RTE	OBQA	HS	BoolQ	ARC-e	ARC-c	Avg
V2-7B	Wanda+LoRA*	73.48	54.51	29.00	51.49	70.98	72.81	39.07	55.91
	Wanda+SPP	73.79	54.15	27.40	51.97	71.80	70.54	37.63	55.32
	Wanda+SQFT	75.61	54.87	29.00	52.86	70.46	72.51	39.07	56.34
	Wanda+SEFT	75.85	56.32	29.80	52.34	67.65	72.60	41.21	**56.54**
	SparseGPT+LoRA*	73.16	53.79	28.60	51.13	77.73	72.34	38.48	56.46
	SparseGPT+SPP	72.69	58.48	28.80	52.63	70.55	69.91	39.16	56.03
	SparseGPT+SQFT	75.37	53.43	30.00	52.38	74.95	72.89	41.38	57.20
	SparseGPT+SEFT	75.53	55.23	30.80	52.99	77.98	72.64	41.13	**58.04**
V3-8B	Wanda+LoRA*	73.87	56.68	27.20	50.69	69.91	71.08	39.51	55.56
	Wanda+SPP	73.32	54.51	27.40	50.37	70.46	70.75	37.97	54.97
	Wanda+SQFT	75.84	60.26	26.40	50.85	77.61	70.24	40.01	57.32
	Wanda+SEFT	76.87	57.76	28.60	52.20	80.83	71.46	41.38	**58.44**
	SparseGPT+LoRA*	79.48	57.04	29.40	53.78	83.30	74.20	43.34	60.07
	SparseGPT+SPP	76.87	57.76	31.20	53.45	80.67	74.12	43.09	59.60
	SparseGPT+SQFT	79.43	63.17	31.40	53.83	82.09	73.19	41.97	60.73
	SparseGPT+SEFT	78.37	62.82	31.20	53.99	83.00	73.40	43.69	**60.92**
V1-13B	Wanda+LoRA*	77.11	59.93	33.60	56.37	82.38	76.56	44.88	61.54
	Wanda+SPP	80.03	62.09	32.60	57.08	76.73	75.88	44.71	61.30
	Wanda+SQFT	79.48	58.84	32.40	56.84	82.54	76.05	46.42	61.79
	Wanda+SEFT	79.32	59.20	33.00	56.58	83.58	77.02	47.44	**62.31**
	SparseGPT+LoRA*	78.37	59.93	31.20	55.23	79.60	76.01	43.26	60.51
	SparseGPT+SPP	80.66	56.68	35.00	57.53	79.78	75.76	45.22	61.52
	SparseGPT+SQFT	79.63	57.28	34.20	57.50	83.45	76.01	44.79	61.84
	SparseGPT+SEFT	80.19	57.40	33.20	56.81	83.52	77.02	45.90	**62.01**

generally achieves competitive or better performance compared to existing baselines, particularly LoRA* and SQFT, across both evaluation metrics. These gains are observed across different model scales and under both pruning strategies.

4.1.2 Supervised fine-tuning

In this section, we evaluate the effectiveness of SEFT on a widely used supervised fine-tuning tasks-Commonsense Reasoning. Table 2 presents the zero-shot accuracy on seven tasks from the commonsense reasoning benchmark. To begin, the evaluation is performed on post-training sparse models generated by Wanda and SparseGPT at a sparsity level of 60%, which provides a balanced trade-off between efficiency and decent performance comparison on this benchmark. The sparse models are then fine-tuned using SEFT and other sparsity-preserving baselines. We include results for three model sizes: LLaMA2-7B, LLaMA3-8B, and LLaMA1-13B, reporting both per task and average performance.

The results show that SEFT consistently outperforms existing sparsity-preserved fine-tuning methods such as SPP and SQFT across all evaluated models and tasks. Compared to LoRA *, SEFT improves the average accuracy by approximately 1–2%, demonstrating its ability to better adapt sparse models to downstream reasoning tasks. In particular, SEFT achieves particularly strong results in OBQA and ARC-c over other baselines. Overall, these findings highlight SEFT's effectiveness in enhancing the overall performance across commonsense reasoning tasks and model sizes.

4.1.3 Computational cost and memory efficiency

In this section, we evaluate the time and memory overhead of different sparsity-preserving fine-tuning methods using the LLaMA-7B model. Since LoRA* includes Wanda pruning post hoc to restore the desired sparsity level, it does not reflect end-to-end training cost and thus is excluded from this analysis. Experiments were conducted with sequence lengths ranging from 256 to 1024 at a rank of 32. Figure 3 (a) presents the memory usage of the different methods across various sequence

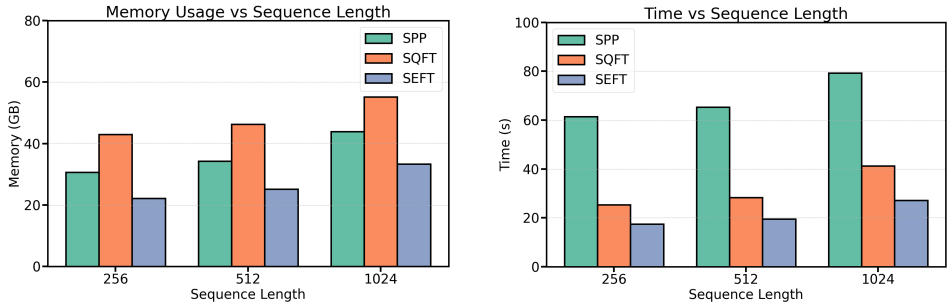

Figure 3: (a) GPU memory usage (in GB) and (b) average training time per step (in seconds) for SEFT and baseline methods on a Commonsense Reasoning task using an H100 GPU. All results are reported without activation checkpointing.

lengths. As the sequence length increases, all methods exhibit higher memory consumption, which is expected. In particular, SEFT consistently uses less memory than other methods in all settings. In particular, SEFT requires approximately half the memory of SQFT.

In Figure 3 (b), we report the average running time per training step, measured over 30 steps on the H100 GPU. The results show that SEFT achieves the fastest training speed across all input lengths, running nearly twice as fast as SQFT and approximately three times faster than SPP. These results underscore the efficiency of our method, demonstrating that SEFT offers a superior balance between performance and resource efficiency when fine-tuning sparse LLMs.

Inference Speedup. We analyze the inference speedup of the sparse LLM achieved after SEFT fine-tuning, as presented in Table 5 in Appendix A. To assess practical gains, we report end-to-end decoding latency using the LLaMA-V2-7B model running in the DeepSparse inference engine [DeepSparse, 2021] on an Intel Xeon Platinum 8360Y CPU with 36 cores. The results show that fine-tuned sparse LLMs yield substantial inference speedups compared to their dense counterparts, achieving up to 2.5× improvement at 70% sparsity. Moreover, the speedup becomes even more pronounced at higher sparsity levels, demonstrating the potential of SEFT to enable both performance efficiency and practical deployment benefits.

4.2 Effect on Other Tasks and Architectures

To further evaluate the generality of SEFT beyond standard benchmarks and the LLaMA-based architectures discussed above, we perform experiments on two challenging benchmarks, MMLU and GSM8K, and extend our evaluation to alternative model architectures, including DeepSeek-7B-chat [Bi et al., 2024] and Mistral-7B-v0.1 [Jiang et al., 2023]. The MMLU benchmark spans a wide range of academic and professional subjects, testing both factual knowledge and reasoning ability, while GSM8K focuses on grade-school math word problems.

Table 3: Performance evaluation of DeepSeek-7B-chat and Mistral-7B-v0.1 models on MMLU and GSM8K benchmark at a sparsity level of 60% by Wanda pruning method. Higher values indicate better performance.

Method		MMLU	GSM8K
DeepSeek-7B-chat	LoRA*	45.57	31.46
	SPP	45.74	31.45
	SQFT	44.23	31.97
	SEFT	**46.14**	**32.52**
Mistral-7B-v0.1	LoRA*	53.52	37.83
	SPP	53.78	39.73
	SQFT	53.57	40.86
	SEFT	**53.97**	**42.91**

Specifically, we apply the Wanda pruning method to compress DeepSeek-7B-chat and Mistral-7B-v0.1 to a sparsity level of 60%, which offers a balanced setting for meaningful performance comparisons. The pruned LLMs are subsequently fine-tuned using SEFT, LoRA*, SPP, and SQFT, and their performance is evaluated on both the MMLU and GSM8K benchmarks. The results show that SEFT consistently outperforms other methods across both tasks and architectures. For instance, on MMLU, SEFT yields competitive performance, while on GSM8K, it achieves up to around 2% gain. These findings underscore the robustness of SEFT in handling diverse tasks and its ability to generalize across different model backbones beyond the LLaMA family. This highlights its potential

as a general-purpose sparse fine-tuning method that is both task- and model-agnostic, making it well-suited for real-world deployment scenarios.

4.3 Effect on N:M Sparsity

In addition to demonstrating the effectiveness of SEFT under unstructured pruning in previous sections, we further evaluate its applicability to structured N:M sparsity patterns, which are increasingly supported by modern hardware (e.g., NVIDIA Ampere and Hopper architectures [NVIDIA, 2021]). To enforce these structured sparsity constraints, the sparse topology evolution phase in SEFT is restricted to exploring and activating updates only within the set of currently active weights ($M_0 = 1$), preserving the required pattern. Table 4 reports the performance of SEFT under 2:4 and 4:8 sparsity configurations on both LLaMA1-7B and LLaMA3-8B models after fine-tuning pruned models using the Wanda method. The results show that SEFT maintains or even improves performance under structured sparsity settings, outperforming other sparsity-preserving baselines. These findings extend the applicability of SEFT beyond unstructured pruning, highlighting its versatility in both algorithmic efficiency and hardware-friendly sparse training.

Table 4: Performance evaluation of LLaMA1-7B and LLaMA3-8B models on LM-eval under different N:M sparsity patterns. Higher values indicate better performance.

Method	**LLaMA1-7B**		**LLaMA3-8B**	
	2:4	4:8	2:4	4:8
LoRA*	57.35	57.71	58.25	60.04
SPP	55.16	56.77	56.55	57.81
SQFT	56.85	57.87	56.99	58.28
SEFT	**58.17**	**58.65**	**59.63**	**60.35**

4.4 Impact of Different Sparsity Levels and Ranks

In Appendix C.1, Table 8 presents the performance evaluation of LLaMA2-7B and LLaMA3-8B models using Wanda pruning under varying sparsity levels. The results show that as the sparsity level increases, SEFT consistently outperforms LoRA* by larger margins. For instance, at a sparsity level of 70%, SEFT achieves significant improvements in both LM-eval and MMLU scores. In particular, on LLaMA3-8B, the relative gain in MMLU reaches approximately 8.69 points. This trend highlights that SEFT's ability to dynamically adapt the sparse topology during fine-tuning is particularly effective compared to maintaining sparsity with post-hoc pruning after fine-tuning in high-sparsity scenarios.

In Appendix C.2, Table 9 illustrates the impact of the number of fine-tuning parameters on the performance of SEFT. As expected, as the number of trainable parameters increases, SEFT shows potential to improve performance. However, SEFT seems to exhibit a stronger upward trend compared to LoRA base method. For instance, while baseline method shows only marginal improvements at high ranks, SEFT demonstrates consistent performance gains, reaching a much higher accuracy of 59.69 at Rank=64 on commonsense reasoning benchmark.

We further explore the impact of various SEFT configurations, including the drop/grow ratio relative to $\tau(t)$ as defined in Eq.5 and Eq.6 (see Appendix C.3), as well as the frequency of sparse topology evolution (Appendix C.4). An ablation study analyzing the contribution of each SEFT component, such as mask constraints during topology evolution (Appendix B.1) and the role of sparsity adaptation (Appendix B.2), is also provided. In addition, we include a sensitivity analysis of different learning rates in Appendix C.5.

5 Related Work

5.1 LLM Pruning

In the era of large language models (LLMs), their enormous sizes pose significant challenges for real-world deployment, including increased computational and memory demands. To address these challenges, recent research has increasingly focused on post-training pruning methods, which start from pre-trained networks and remove redundant parameters to reduce model size and complexity [Ma et al., 2023, Naveed et al., 2023, Zhu et al., 2024]. For instance, SparseGPT utilizes second-order information to solve a layer-wise reconstruction problem, enabling the effective pruning of large models [Frantar and Alistarh, 2023]. Similarly, Wanda introduces a pruning metric that considers

both weight magnitude and corresponding activations [Sun et al., 2023]. More recently, layer-wise prunability has been introduced as a technique to enhance traditional pruning methods by adaptively allocating sparsity across layers based on their individual importance [Yin et al., 2023, Lu et al., 2024b]. However, these methods often struggle to maintain satisfactory performance, particularly in high sparsity regimes. To address this limitation, we propose a novel fine-tuning method for sparse LLMs, enabling the recovery of performance and effective adaptation to downstream tasks.

5.2 Parameter-efficient Fine-tuning

Parameter-Efficient Fine-Tuning (PEFT) has gained significant attention for its ability to adapt large language models (LLMs) to downstream tasks while significantly reducing computational and memory costs [Mangrulkar et al., 2022, Ding et al., 2023, Han et al., 2024]. A notable PEFT approach, Low-Rank Adaptation (LoRA) [Hu et al., 2021], and its variants introduce trainable layer-wise low-rank decomposition matrices into network, allowing efficient parameter updates with minimal overhead [Xu et al., 2023, Zhao et al., 2024, Dettmers et al., 2024]. However, when fine-tuning sparse LLMs, a challenge that arises is that merging dense adapters with sparse weights leads to the overall loss of sparsity, which negates the efficiency benefits of sparse models. Recent studies such as SPP [Lu et al., 2024c] and SQFT [Munoz et al., 2024] have extended LoRA to support sparse LLMs by incorporating masking mechanisms, thereby preserving sparsity during fine-tuning. However, a major limitation of these methods is their reliance on a fixed sparse topology, which restricts their adaptability to diverse downstream tasks.

5.3 Dynamic Sparse Training

Dynamic Sparse Training (DST) is a paradigm that maintains a small fraction of active parameters throughout training by starting with a sparse neural network and dynamically evolving its sparse connectivity using a prune-and-regrow strategy [Mocanu et al., 2018, Evci et al., 2020, Liu et al., 2023, Wu et al., 2025]. It was first well established by [Mocanu et al., 2018] through the Sparse Evolutionary Training (SET) algorithm, which demonstrated superior performance compared to static sparse neural networks by dynamically evolving sparse connectivity during training. Recent advancements have explored diverse pruning criteria, such as magnitude-based [Evci et al., 2020, Liu et al., 2021], weight-balanced [Mocanu et al., 2018], and gradient-based pruning [Yuan et al., 2021], alongside regrowth strategies guided by randomness [Mostafa and Wang, 2019, Xiao et al., 2022], momentum [Dettmers and Zettlemoyer, 2020], and gradient information [Evci et al., 2020]. DST has been effectively applied and widely adopted in various domains, including reinforcement learning [Graesser et al., 2022, Tan et al., 2023], features selection [Sokar et al., 2022, Atashgahi et al., 2022], and image segmentation [Wu et al., 2024]. Building on this foundation, recent work [Ansell et al., 2024] was the first to scale DST to fine-tuning LLMs. Our work extends this further to sparse LLMs by introducing dynamic topology evolution and adaptation, enabling efficient and task-specific fine-tuning for downstream applications.

6 Conclusion

In this work, we propose Sparsity Evolution Fine-Tuning (SEFT), a novel method that dynamically evolves the sparse topology of large language models (LLMs) to better adapt to downstream tasks. By introducing a drop-grow mechanism and enabling the reactivation of previously pruned weights, SEFT supports task-specific adaptation while preserving target sparsity constraints. Extensive experiments across a range of benchmarks demonstrate SEFT's effectiveness in both performance recovery and supervised fine-tuning scenarios. Moreover, SEFT achieves these improvements with significantly lower memory and training time costs. These results underscore its potential as a scalable and efficient solution for fine-tuning sparse LLMs in practical applications.

We provide a discussion of limitations and directions for future work in Appendix F.

Acknowledgements

This work is part of the research program 'MegaMind - Measuring, Gathering, Mining and Integrating Data for Self-management in the Edge of the Electricity System', (partly) financed by the Dutch Research Council (NWO) through the Perspectief program under number P19-25. This work is partly supported by the SmartCHANGE project, funded within EU's Horizon Europe research program (GA No. 101080965). This work used the Dutch national e-infrastructure with the support of the SURF Cooperative, using grant no. EINF-12291.

References

Alan Ansell, Edoardo Ponti, Anna Korhonen, and Ivan Vulić. 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, May 2022.

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

Zahra Atashgahi, Ghada Sokar, Tim van der Lee, Elena Mocanu, Decebal Constantin Mocanu, Raymond Veldhuis, and Mykola Pechenizkiy. Quick and robust feature selection: the strength of energy-efficient sparse training for autoencoders. *Machine Learning*, pages 1–38, 2022.

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024.

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, et al. Lora learns less and forgets less. *arXiv preprint arXiv:2405.09673*, 2024.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

DeepSparse. Neuralmagic deepsparse inference engine. 2021. URL `https://openreview.net/forum?id=z9EkXfvxta`.

Tim Dettmers and Luke Zettlemoyer. Sparse networks from scratch: Faster training without losing performance, 2020.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36, 2024.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235, 2023.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pages 2943–2952. PMLR, 2020.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pages 10323–10337. PMLR, 2023.

Trevor Gale, Matei Zaharia, Cliff Young, and Erich Elsen. Sparse gpu kernels for deep learning. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–14. IEEE, 2020.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, et al. A framework for few-shot language model evaluation. *Version v0. 0.1. Sept*, 10:8–9, 2021.

Laura Graesser, Utku Evci, Erich Elsen, and Pablo Samuel Castro. The state of sparse training in deep reinforcement learning. In *International Conference on Machine Learning*, pages 7766–7792. PMLR, 2022.

Demi Guo, Alexander Rush, and Yoon Kim. 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, August 2021.

Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*, 2024.

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

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Yuxuan Hu, Jing Zhang, Xiaodong Chen, Zhe Zhao, Cuiping Li, and Hong Chen. Lors: Efficient low-rank adaptation for sparse large language model. *arXiv preprint arXiv:2501.08582*, 2025.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Ka-Wei Lee. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. *arXiv preprint arXiv:2304.01933*, 2023.

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. Mistral 7b, 2023. URL `https://arxiv.org/abs/2310.06825`.

N Lee, T Ajanthan, and P Torr. Snip: single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations*. Open Review, 2019.

Pengxiang Li, Lu Yin, Xiaowei Gao, and Shiwei Liu. Owlore: Outlier-weighed layerwise sampled low-rank projection for memory-efficient llm fine-tuning. In *Findings of the Association for Computational Linguistics: ACL 2025*, 2025.

Sean Lie. Cerebras architecture deep dive: First look inside the hardware/software co-design for deep learning. *IEEE Micro*, 43(3):18–30, 2023.

Shiwei Liu, Lu Yin, Decebal Constantin Mocanu, and Mykola Pechenizkiy. Do we actually need dense over-parameterization? in-time over-parameterization in sparse training. In *International Conference on Machine Learning*, pages 6989–7000. PMLR, 2021.

Shiwei Liu, Tianlong Chen, Xiaohan Chen, Xuxi Chen, Qiao Xiao, Boqian Wu, Tommi Kärkkäinen, Mykola Pechenizkiy, Decebal Constantin Mocanu, and Zhangyang Wang. More convnets in the 2020s: Scaling up kernels beyond 51x51 using sparsity. In *ICLR*, 2023. URL `https://openreview.net/pdf?id=bXNl-myZkJl`.

Haiquan Lu, Yefan Zhou, Shiwei Liu, Zhangyang Wang, Michael W Mahoney, and Yaoqing Yang. Alphapruning: Using heavy-tailed self regularization theory for improved layer-wise pruning of large language models. *Advances in Neural Information Processing Systems*, 37:9117–9152, 2024a.

Haiquan Lu, Yefan Zhou, Shiwei Liu, Zhangyang Wang, Michael W Mahoney, and Yaoqing Yang. Alphapruning: Using heavy-tailed self regularization theory for improved layer-wise pruning of large language models. In *Advances in neural information processing systems*, 2024b.

Xudong Lu, Aojun Zhou, Yuhui Xu, Renrui Zhang, Peng Gao, and Hongsheng Li. Spp: Sparsity-preserved parameter-efficient fine-tuning for large language models. In *International Conference on Machine Learning*. PMLR, 2024c.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720, 2023.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and B Bossan. Peft: State-of-the-art parameter-efficient fine-tuning methods. *URL: https://github.com/huggingface/peft*, 2022.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1):2383, 2018.

Hesham Mostafa and Xin Wang. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In *International Conference on Machine Learning*, pages 4646–4655. PMLR, 2019.

Juan Munoz, Jinjie Yuan, and Nilesh Jain. Sqft: Low-cost model adaptation in low-precision sparse foundation models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 12817–12832, 2024.

Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. A comprehensive overview of large language models. *arXiv preprint arXiv:2307.06435*, 2023.

Aleksandra Nowak, Bram Grooten, Decebal Constantin Mocanu, and Jacek Tabor. Fantastic weights and how to find them: Where to prune in dynamic sparse training. *Advances in Neural Information Processing Systems*, 36, 2024.

NVIDIA. Nvidia ampere architecture in-depth, 2021. `https://developer.nvidia.com/blog/nvidia-ampere-architecture-in-depth/`.

Jonas Pfeiffer, Sebastian Ruder, Ivan Vulić, and Edoardo Ponti. Modular deep learning. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL `https://openreview.net/forum?id=z9EkXfvxta`. Survey Certification.

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

Ghada Sokar, Zahra Atashgahi, Mykola Pechenizkiy, and Decebal Constantin Mocanu. Where to pay attention in sparse training for feature selection? *Advances in Neural Information Processing Systems*, 35:1627–1642, 2022.

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

Yiqin Tan, Pihe Hu, Ling Pan, Jiatai Huang, and Longbo Huang. Rlx2: Training a sparse deep reinforcement learning model from scratch. In *The Eleventh International Conference on Learning Representations*, 2023.

Vithursan Thangarasa, Shreyas Saxena, Abhay Gupta, and Sean Lie. Sparse-ift: sparse iso-flop transformations for maximizing training efficiency. In *Proceedings of the 41st International Conference on Machine Learning*, pages 47997–48018, 2024.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

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

Priyan Vaithilingam, Tianyi Zhang, and Elena L Glassman. Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. In *Chi conference on human factors in computing systems extended abstracts*, pages 1–7, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Boqian Wu, Qiao Xiao, Shiwei Liu, Lu Yin, Mykola Pechenizkiy, Decebal Constantin Mocanu, Maurice van Keulen, and Elena Mocanu. E2ENet: Dynamic sparse feature fusion for accurate and efficient 3d medical image segmentation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL `https://openreview.net/forum?id=Xp8qhdmeb4`.

Boqian Wu, Qiao Xiao, Shunxin Wang, Nicola Strisciuglio, Mykola Pechenizkiy, Maurice van Keulen, Decebal Constantin Mocanu, and Elena Mocanu. Dynamic sparse training versus dense training: The unexpected winner in image corruption robustness. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=daUQ7vmGap`.

Xinghao Wu, Xuefeng Liu, Jianwei Niu, Guogang Zhu, and Shaojie Tang. Bold but cautious: Unlocking the potential of personalized federated learning through cautiously aggressive collaboration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19375–19384, 2023.

Qiao Xiao, Boqian Wu, Yu Zhang, Shiwei Liu, Mykola Pechenizkiy, Elena Mocanu, and Decebal Constantin Mocanu. Dynamic sparse network for time series classification: Learning what to "see". In *Advances in Neural Information Processing Systems*, 2022.

Runxin Xu, Fuli Luo, Zhiyuan Zhang, Chuanqi Tan, Baobao Chang, Songfang Huang, and Fei Huang. Raise a child in large language model: Towards effective and generalizable fine-tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9514–9528, November 2021.

Yuhui Xu, Lingxi Xie, Xiaotao Gu, Xin Chen, Heng Chang, Hengheng Zhang, Zhengsu Chen, Xiaopeng Zhang, and Qi Tian. Qa-lora: Quantization-aware low-rank adaptation of large language models. *arXiv preprint arXiv:2309.14717*, 2023.

Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Gen Li, AJAY KUMAR JAISWAL, Mykola Pechenizkiy, Yi Liang, et al. Outlier weighed layerwise sparsity (owl): A missing secret sauce for pruning llms to high sparsity. In *International Conference on Machine Learning*. PMLR, 2023.

Geng Yuan, Xiaolong Ma, Wei Niu, Zhengang Li, Zhenglun Kong, Ning Liu, Yifan Gong, Zheng Zhan, Chaoyang He, Qing Jin, et al. Mest: Accurate and fast memory-economic sparse training framework on the edge. *Advances in Neural Information Processing Systems*, 34:20838–20850, 2021.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient llm training by gradient low-rank projection. *arXiv preprint arXiv:2403.03507*, 2024.

Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. A survey on model compression for large language models. *Transactions of the Association for Computational Linguistics*, 12:1556–1577, 2024.

A Inference Speedup on CPU

While support for unstructured sparsity on modern GPUs remains relatively limited, there is growing attention toward enabling such capabilities. For example, Cerebras has developed specialized hardware designed to support unstructured sparsity at scale [Lie, 2023, Thangarasa et al., 2024]. Additionally, NVIDIA's Ampere and Hopper architectures have introduced hardware-level support for structured sparsity (e.g., 2:4 patterns), enabling modest acceleration through sparse tensor cores [NVIDIA, 2021]. These efforts reflect a broader trend toward making sparsity-aware training and inference feasible on GPU hardware.

Despite these advancements, unstructured sparsity has shown immediate and practical benefits on non-GPU platforms such as CPUs and custom accelerators. For instance, FPGA-based accelerators for sparse RNNs have achieved notable gains in speed and energy efficiency by fully utilizing embedded multipliers. A particularly prominent example is DeepSparse [1], which efficiently deploys large-scale sparse models like BERT on modern Intel CPUs. DeepSparse reports up to 10× model compression with less than 1% accuracy loss, 10× CPU inference speedup with under 2% drop, and as much as 29× speedup with under 7.5% accuracy degradation.

Motivated by these developments, we evaluate actual inference speedup using the DeepSparse engine [DeepSparse, 2021]. Specifically, we measure end-to-end decoding latency for the LLaMA-V2-7B model running on an Intel Xeon Platinum 8360Y CPU with 36 cores. Results show that sparse LLMs fine-tuned with SEFT achieve substantial speedups over their dense counterparts, reaching up to 2.5× improvement at 70% sparsity. Furthermore, the benefit becomes even more pronounced at higher sparsity levels—achieving approximately 4× speedup at 90% sparsity—highlighting the promise of extreme sparsity for efficient inference. These findings underscore the importance of maintaining sparsity throughout the fine-tuning process and point toward the potential of leveraging sparsity-aware GPU operations in future deployment scenarios.

Table 5: End-to-end speedup of LLaMA-V2-7B under different sparsity levels with the DeepSparse inference engine.

Sparsity	**Dense**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**
Latency(ms)	206.36	179.22	113.69	95.18	82.29	63.52	53.64
Throughput	4.84	5.58	8.79	10.50	12.15	15.74	18.64
Speedup	1.0×	1.2×	1.8×	2.2×	2.5×	3.3×	3.9×

B Ablation Study

We conduct a series of ablation studies to better understand the impact of key components in SEFT. Specifically, we analyze (i) the effect of removing the mask constraint during sparse topology evolution, (ii) the role of sparsity adaptation in maintaining the target sparsity level, and (iii) the benefit of using a sensitivity-based pruning criterion over a magnitude-based one.

Table 6: Performance comparison of LLaMA2-7B and LLaMA3-8B models with and without mask constraints for SEFT during sparse topology evolution at different sparsity levels (ρ).

Method		**LLaMA2-7B**		**LLaMA3-8B**	
		$\rho = 0.6$	$\rho = 0.7$	$\rho = 0.6$	$\rho = 0.7$
LM-eval	w. constraint	55.98	48.36	58.17	47.12
	w/o. constraint	**56.54**	**48.87**	**58.44**	**47.44**
MMLU	w. constraint	45.74	31.18	50.03	39.59
	w/o. constraint	**45.95**	**33.51**	**50.33**	**39.87**

[1] https://github.com/neuralmagic/deepsparse

B.1 Effect of Mask Constraints.

In Section 3.1, we introduced how SEFT dynamically evolves the sparse topology during fine-tuning through a drop-grow strategy. Specifically, SEFT enables the delta vector $\boldsymbol{\eta}$ to explore and activate updates for both active weights ($M_0 = 1$) and inactive weights ($M_0 = 0$) in sparse LLMs, enabling dynamic sparse topology evolution. To assess the necessity of reactivating inactive weights, we compare SEFT against a constrained version where the delta vector $\boldsymbol{\eta}$ is restricted to updating only the active weights ($M_0 = 1$) in sparse LLMs. We refer to SEFT as *w/o. constraint* and the constrained version as *w. constraint*.

The results, presented in Table 7, show that for both LLaMA2-7B and LLaMA3-8B models pruned to 60% and 70% sparsity levels using Wanda, SEFT with sparse topology evolution (*w/o. constraint*) consistently outperforms the constrained version on both the commonsense reasoning and MMLU benchmarks. These findings demonstrate that removing the mask constraint, which allows updates to inactive weights, facilitates sparse topology evolution in sparse LLMs. This evolution aligns the model more effectively with task-specific demands, resulting in improved overall performance and mitigating the degradation introduced by pruning methods.

Table 7: Performance comparison of LLaMA1-7B, LLaMA2-7B, and LLaMA3-8B models with and without sparsity adaptation for SEFT during sparse topology evolution at a sparsity level of 60%.

Method		**LM-eval**	**Final sparsity**
LLaMA1-7B	w/o. adapt	58.85	0.596
	w. adapt	58.84	0.600
LLaMA2-7B	w/o. adapt	56.45	0.596
	w. adapt	56.54	0.600
LLaMA3-8B	w/o. adapt	58.60	0.597
	w. adapt	58.44	0.600

B.2 Effect of Sparsity Adaption.

In Section 3.2, we proposed a sparsity adaptation process to restore the model to the desired sparsity level, as sparse topology evolution in SEFT without a mask constraint may result in a denser model by reactivating previously inactive weights. We conducted experiments to evaluate the effectiveness of sparsity adaptation on LLaMA1-7B, LLaMA2-7B, and LLaMA3-8B models pruned to 60% sparsity using Wanda, comparing SEFT with and without sparsity adaptation. As shown in Table 7, the results indicate that without sparsity adaptation, the sparse LLMs become slightly denser after fine-tuning, with a sparsity level of approximately 59.6%. In contrast, with sparsity adaptation, the models maintain their original sparsity levels while achieving comparable performance.

Furthermore, in Figure 5 (a), we evaluate the effectiveness of sensitivity-based criterion by comparing it to the commonly used magnitude-based criterion for pruning back to the target sparsity. Magnitude-based criterion are widely used in dynamic sparse training (DST) scenarios. Our results demonstrate that sensitivity-based criterion consistently outperform magnitude-based criterion. This advantage is likely due to the sensitivity-based approach accounting not only for the magnitude of the weights but also for gradient information, which reflects their future importance during training.

C Sensitivity Analysis

We conduct a detailed sensitivity analysis to examine how SEFT responds to different hyperparameter choices. Specifically, we study (i) the impact of sparsity levels on fine-tuning performance, (ii) the effect of the number of trainable parameters, (iii) the influence of drop rate and evolution frequency in sparse topology evolution, and (iv) the sensitivity to learning rate configurations. These analyses offer insights into SEFT's robustness and adaptability across settings, and help guide optimal hyperparameter choices for various deployment scenarios.

C.1 Impact of Sparsity Level

Table 8 presents the performance evaluation of LLaMA2-7B and LLaMA3-8B models using Wanda pruning under varying sparsity levels. The results show that as the sparsity level increases, SEFT consistently outperforms LoRA* by larger margins. For instance, at a sparsity level of 70%, SEFT achieves significant improvements in both LM-eval and MMLU scores compared to LoRA*. Notably, on LLaMA3-8B, the relative gain in MMLU reaches approximately 8.69 points (39.87 vs. 31.18). This trend highlights that SEFT's ability to dynamically adapt the sparse topology during fine-tuning is particularly effective in high-sparsity scenarios. The primary reason for this is that post-training pruning methods like Wanda tend to degrade more significantly at higher sparsity levels, resulting in greater performance gaps. By leveraging sensitivity-based criterion, SEFT is able to better identify and optimize critical parameters, maintaining strong performance even under extreme sparsity conditions where LoRA* struggles to achieve comparable results.

Table 8: Performance comparison of LLaMA2-7B and LLaMA3-8B models on LM-eval and MMLU under different sparsity levels (ρ).

Method		**LLaMA2-7B**			**LLaMA3-8B**		
		ρ=0.5	ρ=0.6	ρ=0.7	ρ=0.5	ρ=0.6	ρ=0.7
LM-eval	LoRA*	**60.12**	56.19	46.36	62.90	55.56	44.82
	SEFT	59.30	**56.54**	**48.87**	**63.04**	**58.44**	**47.44**
MMLU	LoRA*	48.21	42.29	25.37	57.40	49.18	31.18
	SEFT	**48.90**	**45.95**	**33.51**	**57.51**	**50.33**	**39.87**

Table 9: Performance comparison of LLaMA1-7B fine-tuned on the commonsense reasoning dataset at a sparsity level of 60%. The results compare different numbers of fine-tuning parameters and report the average zero-shot accuracy across seven tasks from the commonsense reasoning benchmark.

Method	Rank=8	Rank=16	Rank=32	Rank=64
LoRA*	56.81	57.13	57.32	57.17
SEFT	57.74	57.61	58.84	59.69

C.2 Impact of Number of Fine-Tuning Parameters

In this section, we analyze how the number of fine-tuning parameters impacts the performance of SEFT and LoRA* methods. The comparison is based on different numbers of fine-tuning parameters, determined by the ranks in LoRA. For SEFT, the number of fine-tuning parameters is aligned with the corresponding parameter count of LoRA ranks to ensure a fair comparison.

As shown in Table 9, the performance of both LoRA* and SEFT improves as the number of fine-tuning parameters increases, which is expected. However, SEFT exhibits a significantly stronger upward trend compared to LoRA*. For instance, while LoRA* shows only marginal improvements at higher ranks (e.g., from 57.32 at Rank=32 to 57.17 at Rank=64), SEFT demonstrates consistent performance gains, reaching a much higher accuracy of 59.69 at Rank=64.

This trend indicates that SEFT is better equipped to utilize the additional fine-tuning capacity to adapt sparse LLMs to downstream tasks. By dynamically evolving the sparse topology during fine-tuning, SEFT effectively redistributes its parameter budget toward critical updates, enabling it to achieve more substantial improvements as the number of parameters increases. Overall, SEFT consistently outperforms LoRA* across all ranks, with the performance gap widening at higher parameter counts. These results highlight SEFT's scalability and its ability to efficiently leverage additional fine-tuning parameters for enhanced task-specific performance.

C.3 Impact of Drop Rate

To investigate the effect of varying drop rates, which determine the fraction of weights pruned during each update, we conducted experiments using different initial drop rates for SEFT fine-tuning on the

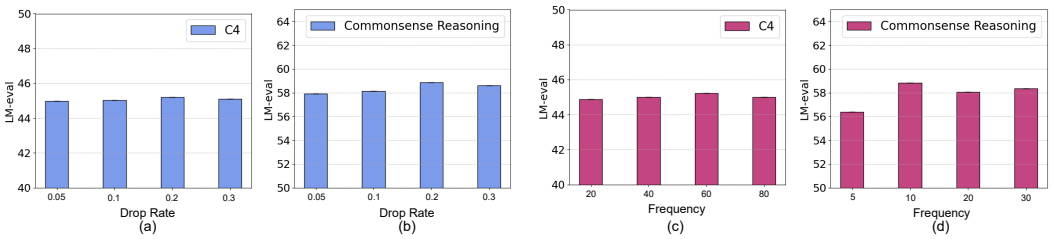

Figure 4: Impact of drop rate during sparse topology evolution on LM-eval evaluation of LLaMA1-7B fine-tuning for (a) C4 and (b) commonsense reasoning datasets. Furthermore, the impact of update frequency during sparse topology evolution is evaluated on LM-eval for LLaMA1-7B fine-tuning on (c) C4 and (d) commonsense reasoning datasets.

C4 and Commonsense Reasoning datasets. Drop rates of 0.05, 0.1, 0.2, 0.3 were evaluated, along with a cosine schedule decay for the drop/grow ratio, as proposed in [Evci et al., 2020].

As shown in Figure 4 (a) (b), the results demonstrate that a drop rate of 0.2 consistently achieves the best performance across these tasks. Lower drop rates, such as 0.05, fail to sufficiently adapt the sparse topology, limiting performance improvements, while higher drop rates, such as 0.3, can overly perturb the model, disrupting training convergence. The balanced drop rate of 0.2 was adopted as the default setting for all subsequent experiments, as it effectively balances adaptation and stability during sparse topology evolution.

C.4 Impact of Update Frequency k

The frequency of sparse topology evolution, which determines how often the sparse topology is updated during training, is another critical factor. We evaluated different drop/grow frequencies for fine-tuning on the C4 dataset (10, 20, 40, 80 steps) and the Commonsense Reasoning dataset (5, 10, 20, 30 steps).

As shown in Figure 4 (c) (d), the optimal frequency varies by task, reflecting differences in training dynamics and the rate at which sparse topology adaptation benefits performance. For example, on the C4 dataset, the best performance was achieved with an update frequency of 60 steps, while for the Commonsense Reasoning dataset, the optimal frequency was 10 steps. These frequencies were adopted as the default settings in our main experiments.

The results also reveal that overly frequent updates can introduce excessive instability, preventing the model from converging effectively. Conversely, infrequent updates reduce the model's ability to adapt its sparse topology, thereby limiting the benefits of sparsity. Striking the right balance in the frequency of topology evolution is crucial for maximizing SEFT's performance across diverse tasks.

C.5 Impact of Learning Rate

In this section, we analyze how the learning rate affects the performance of the sparse fine-tuning method SEFT and the LoRA-based fine-tuning method LoRA*. As shown in Figure 5 (b), we conduct experiments on LLaMA1-7B under a sparsity level of 0.6, using the commonsense reasoning dataset. The results report the zero-shot accuracy across seven tasks from the commonsense reasoning benchmark. From the figure, we observe that LoRA* generally performs better with a larger learning rate, achieving the best performance at a learning rate of 2e-3. In contrast, SEFT performs best at a slightly smaller learning rate of 1e-3. Interestingly, under smaller learning rates, SEFT achieves more significant improvements compared to LoRA*.

This phenomenon may be attributed to the fundamental differences in how the two methods update sparse LLMs. SEFT directly updates the original weights of the sparse LLMs, enabling more precise control and better adaptation, especially under smaller updates at lower learning rates. This behavior is similar to dense full fine-tuning, where the original model weights are explicitly updated to reflect task-specific requirements. In contrast, LoRA-based fine-tuning approximates updates through low-rank matrices, indirectly influencing the original weights of the sparse LLMs. This indirect update

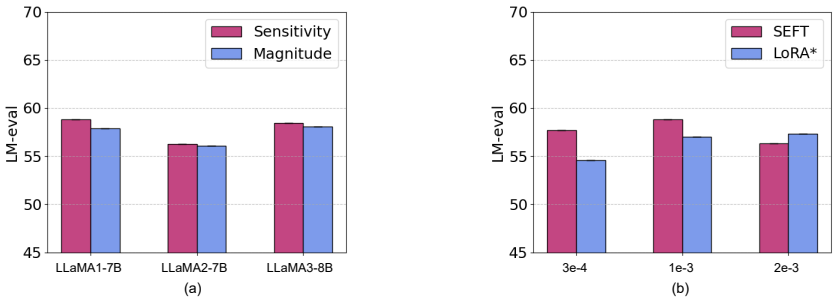

Figure 5: (a) LM-eval comparison of LLaMA1-7B, LLaMA2-7B, and LLaMA3-8B models with sensitivity-based and magnitude-based sparsity adaptation in SEFT fine-tuning at 60% sparsity using Wanda pruning. (b) LM-eval comparison of LLaMA1-7B models at 60% sparsity using Wanda pruning for LoRA* and SEFT fine-tuning across different learning rates.

mechanism allows for more flexible and larger updates, as the original weights remain intact and act as a safeguard against errors introduced by the approximation.

These results highlight the critical role of learning rate selection in optimizing the performance of sparse fine-tuning methods. To ensure a fair comparison, we performed a grid search for both fine-tuning approaches.

D Peformance Comparison with Original LoRA

In the main paper, to ensure a fair comparison, all fine-tuned LLMs were restored to the same target sparsity level after fine-tuning. In this section, we present the original results for LoRA, which do not include sparsity restoration. In other words, the fine-tuned LLMs remain dense after merging the low-rank matrices with the original model weights.

Tables 11 and 10 provide a detailed comparison between SEFT and LoRA across three datasets: C4, Commonsense Reasoning. The results demonstrate that SEFT achieves comparable performance to LoRA, even when the latter uses dense connections in the LLMs. However, SEFT maintains the sparse topology at the original sparsity level, preserving the efficiency of the sparse model.

These findings underscore the effectiveness and efficiency of SEFT, as it not only matches the performance of LoRA fine-tuning with dense models but also retains the computational advantages of sparsity, making it a more practical solution for sparse fine-tuning of large language models, , as discussed in detail in Section A.

Table 10: Performance comparison of different fine-tuning methods applied to various sparse LLaMA models at a sparsity level of ($\rho = 0.6$). Results include the average accuracy of zero-shot evaluation across seven tasks from the commonsense reasoning benchmark.

LLaMA	Method	LM-eval
V2-7B	Wanda+LoRA	56.66
	Wanda+SEFT	56.54
	SparseGPT+LoRA	58.24
	SparseGPT+SEFT	58.04
V3-8B	Wanda+LoRA	59.51
	Wanda+SEFT	58.44
	SparseGPT+LoRA	60.66
	SparseGPT+SEFT	60.92
V1-13B	Wanda+LoRA	61.09
	Wanda+SEFT	62.31
	SparseGPT+LoRA	61.99
	SparseGPT+SEFT	62.01

E Discussion

Preserving sparsity during and after the fine-tuning of sparse LLMs is essential for maintaining computational efficiency, particularly in resource-constrained scenarios. Methods like LoRA, originally designed for resource-efficient fine-tuning of dense pre-trained models, fail to preserve sparsity after merging, limiting their effectiveness for sparse LLMs finetuning. Recent approaches, such as SPP

Table 11: Performance comparison of different methods with and without fine-tuning applied to sparse LLaMA models at a sparsity level of ($\rho = 0.7$). We provide average accuracies of 7 zero-shot tasks together with PPL (lower is better).

LLaMA	Method	PPL (↓)	LM-eval (↑)
V2-7B	Wanda+LoRA	10.82	45.08
	Wanda+SEFT	11.19	45.61
	SparseGPT+LoRA	10.53	47.80
	SparseGPT+SEFT	11.00	47.95
V3-8B	Wanda+LoRA	16.12	44.76
	Wanda+SEFT	16.17	44.55
	SparseGPT+LoRA	17.88	47.19
	SparseGPT+SEFT	15.09	48.89

[Lu et al., 2024c], extend LoRA to support sparse LLMs by incorporating masking mechanisms to preserve sparsity. However, SPP enforces a fixed sparse topology across all tasks, which limits its adaptability to the specific requirements of downstream applications.

Sparse fine-tuning was first scaled to dense LLM fine-tuning in SpIEL [Ansell et al., 2024]. Unlike LoRA-based methods, which use trainable low-rank matrices to parameterize adaptations, fine-tuning directly updates a small fraction of model weights through updates and their corresponding index vectors. In this paper, we extend this concept to sparse LLMs fine-tuning by enabling the updates to dynamically evolve and adapt in sparse LLMs. While SEFT builds upon the core concept of sparse fine-tuning, it introduces several key innovations tailored to sparse LLMs:

(1) Sparse setting. Unlike SpIEL, which operates on dense models, SEFT is designed for fine-tuning already-pruned sparse LLMs. This setting imposes strict sparsity constraints that require careful management during training.

(2) Update flexibility sparsity adaptation. SEFT allows updates to zero-valued (previously pruned) weights, enabling recovery of important connections for downstream tasks. It also incorporates a sparsity adaptation step to ensure the model maintains its target sparsity after each update cycle—both of which are absent in SpIEL.

These distinctions make SEFT a more tailored solution for fine-tuning sparse LLMs across varying sparsity levels.

F Limitations and Future Work

While our work highlights the effectiveness of SEFT in fine-tuning sparse LLMs and enhancing their performance on downstream tasks, there are certain limitations that warrant further investigation.

One key limitation is the need to compute the full gradient during the sparse topology evolution phase, where the sparse delta gradient is collected based on the gradient values. While this approach ensures the dynamic adjustment of the sparse topology, it does not fully exploit the potential computational efficiency gains associated with sparsity, particularly on GPUs. The current implementation involves calculating the entire dense gradient matrix and then extracting the required sparse updates, which is suboptimal for leveraging the inherent benefits of sparsity in terms of speed and resource utilization.

Writing an efficient CUDA kernel tailored for dynamic sparse training operations is an area of ongoing work. Such a kernel would enable direct computation of sparse updates without requiring operations on the full gradient matrix, significantly reducing memory usage and computational overhead. Once optimized, this approach could fully unlock the potential of SEFT and dynamic sparse training paradigms, making them more practical and efficient for large-scale federated and distributed learning scenarios.

G Impact Statements

This paper contributes to the development of efficient fine-tuning methods for large language models (LLMs), addressing the challenges posed by their substantial computational and performance demands and enhancing their feasibility for real-world deployment. While our contributions do not inherently lead to negative societal impacts, we encourage the community to remain mindful of potential ethical and practical implications when extending or applying our research.

H Experimental Settings

For training 7B models, we use a learning rate of 1e-3, except for Mistral-7B, where we use 5e-5. For 8B models, we adopt a learning rate of 3e-4. Gradient accumulation steps are set to 128, and we apply cosine learning rate decay throughout training. We use the AdamW optimizer with the default settings provided by the Transformers library, and no weight decay is applied. For baseline methods, we primarily build upon the official implementations of SPP[2] and SQFT[3].

All models are fine-tuned using post-training pruned LLMs obtained via SparseGPT and Wanda under various sparsity patterns and levels. We use fixed hyperparameters across experiments and do not perform tuning for specific sparsity configurations. We follow the dataset configurations in [Li et al., 2025], with details summarized in Table 12.

Table 12: Hyperparamters used of SEFT for fine-tuning on various benchmarks.

Benchmarks	Commonsense Reasoning	MMLU	GSM8K
Train Samples	170K	99.8K	7.4K
Test Samples	22.4K	14K	1.3K
Max Length	512	512	512
Training Epoch	1	1	5
Drop Rate	0.2	0.2	0.2
Frequency	10	60	10

[2]https://github.com/Lucky-Lance/SPP

[3]https://github.com/IntelLabs/Hardware-Aware-Automated-Machine-Learning