EBFT: Effective and Block-Wise Fine-Tuning for Sparse LLMs

Anonymous ACL submission

Abstract

Existing methods for fine-tuning sparse LLMs often suffer from resource-intensive requirements and high retraining costs. Additionally, many fine-tuning methods often rely on approximations or heuristic optimization strategies, which may lead to suboptimal solutions. To address these issues, we propose an efficient and fast framework for fine-tuning sparse LLMs based on minimizing reconstruction error. Our approach involves sampling a small dataset for calibration and utilizing backpropagation to iteratively optimize block-wise reconstruction error, on a block-by-block basis, aiming for optimal solutions. Extensive experiments on various benchmarks consistently demonstrate the superiority of our method over other baselines. For instance, on the Wikitext2 dataset with LlamaV1-7B at 70% sparsity, our proposed EBFT achieves a perplexity of **16.88**, surpassing the state-of-the-art DSnoT with a perplexity of 75.14. Moreover, with a structured sparsity ratio of 26%, EBFT achieves a perplexity of **16.27**, outperforming LoRA (perplexity 16.44). Furthermore, the fine-tuning process of EBFT for LlamaV1-7B only takes approximately **30** minutes, and the entire framework can be executed on a single **16GB** GPU. The source code is available at https://github.com/anonymousACL2024/EBFT.

1 Introduction

LLMs have demonstrated remarkable potential in various NLP tasks. However, the large sizes of these models pose challenges in terms of resource requirements for deployment. For instance, the inference of GPT-3 (Brown et al., 2020) in half-precision floating-point format demands at least 5 80G A100 GPUs. To address this issue, several model compression methods, such as network quantization (Lin et al., 2023; Frantar et al., 2022), network pruning (Frantar and Alistarh, 2023), and knowledge distillation (Hsieh et al., 2023), have been proposed to compress and accelerate these Large Language Models. Among these methods, network pruning has gained increasing attention. However, pruning often leads to a decline in the performance of sparse models. To address this issue, recent works (Zhang et al., 2023d; Frantar and Alistarh, 2023; Zhang et al., 2023a) have emerged that can fine-tune the pruned models to recover their performance through regression reconstruction, costly retraining, or other heuristic methods. In this paper, we introduce EBFT, a framework designed to effectively fine-tune sparse LLMs, significantly enhancing the performance and generality of pruned models.

Dataset used for fine-tuning. Some existing pruning then fine-tuning approaches require significant retraining resources, partly due to the large size of the retraining dataset. For example, LLM-Pruner (Ma et al., 2023) employs Alpaca-cleaned (Taori et al., 2023) as its fine-tuning dataset to restore the performance of sparse LLMs. Alpaca-cleaned consists of 51.8K rows of data, resulting in substantial time costs for fine-tuning LLMs. Similarly, Sheared Llama (Xia et al., 2023) employs RedPajama (Computer, 2023), containing 2.17M rows of data, for LLM pruning and fine-tuning, which incurs huge resource costs. In this paper, we sample a small calibration dataset comprising only 256 1024-token segments extracted from C4 (Raffel et al., 2020). By fine-tuning sparse LLMs using these samples, we effectively reduce the resource requirements and time costs associated with the process.

Optimization algorithm. Current LLM pruning methods, like SparseGPT (Frantar and Alistarh, 2023), construct reconstruction errors based on feature maps before and after pruning. They approximate the reconstruction error using the second-order term of Taylor's Formula and optimize it by regression reconstruction. Wanda (Sun et al., 2023)

can be viewed as an approximation of the pruning criteria used in SparseGPT. DSnoT (Zhang et al., 2023d) utilizes masks from Wanda or SparseGPT as initialization and designs a heuristic criterion to reselect masks that can reduce the reconstruction error further. These algorithms only optimize an approximation and often rely on heuristic experiences, leading to sub-optimal solutions. In contrast, our method defines the block-wise reconstruction error and directly optimizes it through backpropagation (Werbos, 1990), ensuring the attainment of an optimal and convergent solution.

Fine-tuning costs. EBFT can be integrated with any pruning method and optimizes the block-wise reconstruction error through a backpropagation algorithm. Our framework can avoid the simultaneous loading of all LLM blocks onto the GPU and require only a few samples, significantly reducing costs. Experimental results indicate that the time required of EBFT for fine-tuning each block in Llama-7B (Touvron et al., 2023a) ranges between 50 and 60 seconds, resulting in a total time cost of approximately 30 minutes. EBFT enables fine-tuning Llama-7B with a single 16GB GPU, making LLM fine-tuning feasible even under resource-constrained conditions.

In summary, our **contributions** can be summarized as follows:

- We introduce EBFT, a block-by-block fine-tuning framework for sparse LLMs, which requires only a few samples, significantly reducing resource dependencies.
- EBFT updates the network based on the minimization of block-wise reconstruction error through backpropagation, resulting in an optimal and convergent solution.
- EBFT consistently surpass other state-of-the-art algorithms on various benchmarks and models, demonstrating the strong efficiency of our method.

2 Related Work

Network pruning. According to different levels of granularity, pruning can be categorized into unstructured pruning, structured pruning, and semi-structured pruning. **(1) Unstructured pruning**. Unstructured pruning methods involve removing individual weights in the weight matrix. Han et al. (Han et al., 2015) proposed an algorithm based on l_1 and l_2 regulation, suggesting that smaller-norm weights are less important. LTH (Frankle and Carbin, 2018) increases the sparsity ratio during training and utilizes magnitude for pruning.**(2) Structured pruning**. Structured pruning involves removing entire rows or columns of the weight matrix. Li et al. (Li et al., 2016) use the l_l-norm as the importance scores for channels. A pruning method (Sanh et al., 2020) called movement pruning was proposed, which used the product of weight value and its gradient as the criterion for importance, surpassing magnitude pruning on BERT (Devlin et al., 2018). Cofipruning (Xia et al., 2022) generates masks for BERT pruning via l0 regularization (Louizos et al., 2017; Wang et al., 2019). Guo et al. (Guo et al., 2023b) analyze existing pruning criteria and propose a method based on the information bottleneck principle (Tishby et al., 2000; Tishby and Zaslavsky, 2015).**(3) Semi-structured pruning**. Semi-structured pruning, also known as N:M sparsity (Zhou et al., 2021; Zhang et al., 2022), ensures that for every continuous M weights in the weight matrix, only N weights are non-zero. N:M sparsity can accelerate the sparse model on specific devices. Zhang et al. (Zhang et al., 2023c) proposed transposable (Hubara et al., 2021) bi-directional masks to accelerate sparse models in both the forward and backward processes.

Fine-tuning for pruned LLMs. For LLMs, specific pruning methods (Ashkboos et al., 2024; An et al., 2023; Syed et al., 2023; Li et al., 2023) have been proposed. LoraPruner (Zhang et al., 2023a), LLM-pruner (Ma et al., 2023), and Compresso (Guo et al., 2023a) aim to remove entire attention heads or FFN units in the transformers (Vaswani et al., 2017), followed by fine-tuning on a large dataset using PEFT (Hu et al., 2021). However, these methods suffer from performance degradation and high retraining costs. SparseGPT (Frantar and Alistarh, 2023) employs OBS (Hassibi et al., 1993) to prune the weights of LLMs and recovers their performance through regression reconstruction. Wanda (Sun et al., 2023) proposes a new importance criterion, which approximates the criterion used in SparseGPT. DSnoT (Zhang et al., 2023d) aims to fine-tune sparse LLMs and designs a criterion to further reduce reconstruction error by reselecting masks. These methods require costly retraining or rely on approximation and heuristic optimization strategies, resulting in significant resource consumption or sub-optimal solutions.To address these challenges, we propose a fine-tuning

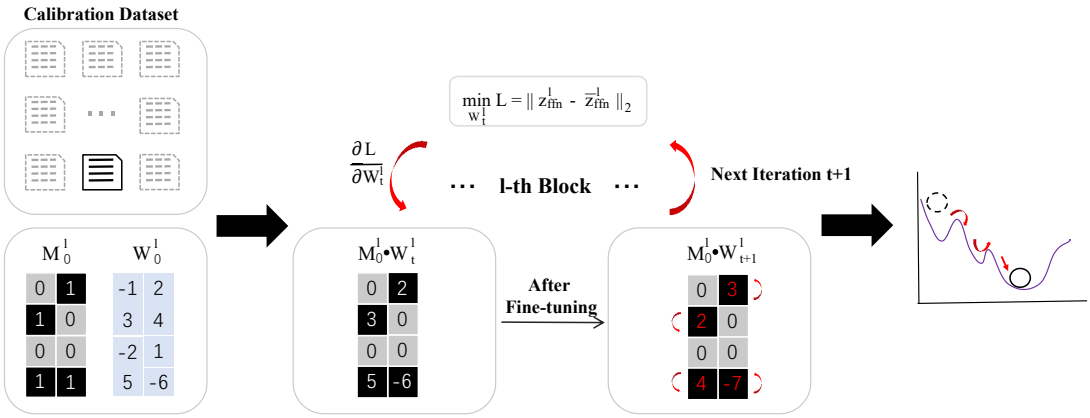

Figure 1: EBFT can be integrated with any other pruning methods, requiring only a small number of samples from C4. When the initial mask M_0^l and weight W_0^l are provided, EBFT updates the weight W_t^l through backpropagation to optimize the reconstruction error L mentioned in Eq.4, ultimately achieving a convergent and optimal solution. Here, W_t^l represents the weight vector of the l-th block of the LLM in the t-th iteration.

framework called EBFT, which helps us obtain an optimal and convergent sparse model.

3 Methodology

3.1 Preliminaries

Large Language Model. The structure of a large language model is based on the transformer, which consists of multiple stacked blocks. Each block consists of two modules: multi-head self-attention (MHA) and multi-layer perceptron (MLP). MHA typically comprises four linear layers, while MLP consists of two or three linear layers. For the l-th block in the large language model, it can be formulated as follows:

$$\begin{aligned} z_{attn}^l &= MHA(W_{mha}^l, LN(z_{ffn}^{l-1})) + z_{ffn}^{l-1}, \\ z_{ffn}^l &= MLP(W_{mlp}^l, LN(z_{attn}^l)) + z_{attn}^l, \end{aligned} \tag{1}$$

where W_{mha}^l represents the weight vector of the multi-head self-attention module, and W_{mlp}^l represents the weight vector of the multi-layer perceptron module in the i-th block. LN represents the layer normalization function. z_{ffn}^{l-1} denotes the output of the (l-1)-th block, which serves as the input to the l-th block. The input z_{ffn}^{l-1} is first passed through the MHA module and then through the MLP module.

Pruning for LLMs. Existing pruning methods for LLMs (Frantar and Alistarh, 2023; Zhang et al., 2023d; Boža, 2024; Das et al., 2023) typically employ the reconstruction error of the layer-wise feature maps before and after pruning as the optimization objective. This objective can be defined as follows:

$$\min_{M,\bar{W}} ||WX - (M \odot \bar{W})X||_2, \ s.t. \ 1 - \frac{||M||_0}{N} = S, \tag{2}$$

where X represents the input activation. W and $\bar{W}$ represent the original and remaining weight vectors, respectively, of any layer in the block of the LLM. M$\in \{0,1\}^N$ is the mask for this layer, indicating whether the corresponding weights should be preserved (1) or discarded (0). S is the pre-designed target sparsity, and N denotes the total number of weights in the layer.

These methods often employ the second-order term of the Taylor formula to approximate the layer-wise reconstruction error in Eq. 2 or design heuristic criteria to optimize Eq. 2. However, these approaches may result in suboptimal solutions.

3.2 EBFT

Overview. We propose a framework called EBFT for the fine-tuning of sparse LLMs, aiming to achieve optimal solutions. Unlike other costly methods that involve pruning and then fine-tuning on a large dataset (Xia et al., 2023; Ma et al., 2023; Zhang et al., 2023a), EBFT only requires a small calibration dataset consisting of a few samples. Specifically, we extract 256 1024-token samples from C4 and use them as the calibration dataset denoted as D_c. The principle of EBFT is based on minimizing the block-wise reconstruction error. An overview of our algorithm is depicted in Fig. 1.

Optimization objective. For the l-th block in

the sparse LLM, it can be formulated as:

$$\bar{z}^l_{attn} = MHA(\bar{W}^l_{mha}, LN(\bar{z}^{l-1}_{ffn})) + \bar{z}^{l-1}_{ffn},$$
$$\bar{z}^l_{ffn} = MLP(\bar{W}^l_{mlp}, LN(\bar{z}^l_{attn})) + \bar{z}^l_{attn}, \quad (3)$$

where $\bar{W}^l_{mha} = M^l_{mha} * W^l_{mha}$ and $\bar{W}^l_{mlp} = M^l_{mlp} * W^l_{mlp}$ represent the remain weight vector of the multi-head self-attention module and multi-layer perceptron module, respectively, in the l-th block. M^l_{mha} and M^l_{mlp} represent their corresponding masks. $\bar{z}^l_{ffn}$ denotes the output of the l-th block after pruning.

we define our block-wise optimization objective as:

$$\min_{\bar{W}^l_{mha}, \bar{W}^l_{mlp}} ||z^l_{ffn} - \bar{z}^l_{ffn}||_2 \quad (4)$$

In Eq. 4, we preserve the masks obtained from other pruning methods unchanged and focus on optimizing the remaining weights within the current block.

Compared to the layer-wise reconstruction error in Eq.2, the block-wise optimization process in Eq.4 allows for interaction and information exchange among different layers within the block. This enables the model to avoid potential issues associated with local optima in layer-wise optimization and explore the solution space more effectively, leading to the discovery of a globally optimal solution. Our EBFT is to directly optimize Eq.4 without relying on any approximations or heuristic methods.

Optimization algorithm. Unlike some methods (Kwon et al., 2022; Frantar and Alistarh, 2023; Zhang et al., 2023a) that update the weights of LLM based on regression reconstruction or costly retraining, we employ the backpropagation algorithm to minimize Eq. 4 by updating the value of the variable $\bar{W}^l_{mha}$ and $\bar{W}^l_{mlp}$ block by block on the D_c, without utilizing any heuristic methods.

The workflow of our EBFT framework is illustrated in Alg. 1. Prior to the fine-tuning process, we establish a maximum iteration T to control the overall fine-tuning cost. Specifically, we set T to 10 epochs. During the fine-tuning phase, if the loss remains unchanged or changes within a small range, we consider the loss to have converged. At this point, the fine-tuning algorithm for the current block will terminate early, allowing us to proceed to the subsequent block for a new round of fine-tuning.

In Alg. 1, m_0 can be obtained from any pruning methods. α represents the learning rate which determines the size of updating step for the variable $\bar{W}^t_l$. Specifically, we set the learning rate to 2e-4.

Algorithm 1: Pseudocode of EBFT

input : sparse LLM F with L blocks; Initial Mask m_0; Calibration dataset D_c; Max fine-tuning iterations T; Learning rate α;
output : Fine-tuned sparse LLM F_T
for *block* $l = 1$ **to** L **do**
  for *iteration* $t = 0$ **to** T **do**
    $E \leftarrow$ Calculating the reconstruction error via Eq. 4.
    If E is **convergent:**
      break
    $\nabla \bar{W}^l_t \leftarrow$ Calculating the gradient of $\bar{W}^l_t$ with respect to E through Bp algorithm.
    $\bar{W}^l_{t+1} \leftarrow \bar{W}^l_t$ -$\alpha \nabla \bar{W}^l_t$
  end
end
return F_T;

4 Experiments

Models and Baselines. We apply magnitude pruning, SparseGPT, and Wanda techniques to the widely adopted LLMs, LlamaV1 (Touvron et al., 2023a) and LlamaV2 (Touvron et al., 2023b). Subsequently, we compare the evaluation results of the state-of-the-art method DsnoT (Zhang et al., 2023d) with our approach on the pruned LlamaV1 and LlamaV2, considering both unstructured sparsity and N:M sparsity. To further assess the effectiveness of our method, we also compare EBFT with LoRA (Hu et al., 2021) under structured sparsity using FLAP (An et al., 2023).

Evaluation. To evaluate the performance of our method and other baselines, we conduct comparisons on the widely-used dataset Wikitext2 (Merity et al., 2016) to calculate perplexity scores. Additionally, we perform a series of zero-shot tasks, including PIQA (Bisk et al., 2020), StoryCloze (Mostafazadeh et al., 2017), ARC-Easy and ARC-Challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), Winogrande (Sakaguchi et al., 2021), and Boolq (Clark et al., 2019). These tasks aim to assess the generality of the pruned model.

4.1 Language Modeling

Unstructured Pruning. We perform comprehensive comparative experiments on the Wikitext2 dataset, and the results are presented in Table.1 We compare the perplexity of pruned LlamaV1 and LlamaV2 models using our method, DsnoT, magnitude pruning, Wanda, and SparseGPT across a range of sparsity levels, from 50% to 90%. The

Method \ Sparsity	LlamaV1-7B 50%	60%	70%	80%	90%	LlamaV2-7B 50%	60%	70%	80%	90%
Magnitude	17.29	559.99	48415	132176	317879	16.03	1924.81	49906	nan	nan
w. DsnoT	13.80	127.67	9614795	37474	202562	13.90	3749.55	14271e4	21760e2	34462e2
w. Ours	**7.11**	**9.53**	**26.30**	**659.12**	**9718.99**	**6.59**	**9.29**	**33.50**	**462.32**	**2930.51**
Wanda	7.26	10.69	88.84	5671.52	12748	6.94	10.96	78.26	3136.23	6995.88
w. DsnoT	7.14	10.40	75.14	3635.94	9043.63	6.85	10.85	75.55	4197.74	7311.58
w. Ours	**6.81**	**8.59**	**16.88**	**118.38**	**2993.32**	**6.18**	**7.90**	**16.94**	**72.80**	**903.45**
SparseGPT	7.20	10.40	27.00	167.55	3912.78	7.09	10.54	29.37	131.17	1542.22
w. DsnoT	9.25	9.68	46.99	8038.14	198898	6.97	10.23	59.62	2510.54	49639
w. Ours	**6.73**	**8.33**	**16.07**	**141.15**	**3366.39**	**6.20**	**7.88**	**18.13**	**130.89**	**1233.80**

Table 1: Comparison of perplexity for pruning and fine-tuning LlamaV1-7B and LlamaV2-7B on Wikitext2 dataset at unstructured sparsity levels ranging from 50% to 90%.

experimental results show the strong effectiveness of our EBFT. We can observe that regardless of the magnitude pruning method used, be it SparseGPT or Wanda, our method enhances the performance of the sparse model. For instance, with magnitude pruning, our method achieves a perplexity of 7.11, surpassing the perplexity of 17.29 before fine-tuning, and even outperforming Wanda (7.26) and SparseGPT (7.20).

We also find that as the sparsity increases, two observations emerge: (1) The state-of-the-art DsnoT loses its effectiveness as a fine-tuning method. For example, when using SparseGPT, DsnoT degrades the performance of the sparse model at sparsity levels of 70%, 80%, and 90%. This demonstrates the limitations of heuristic optimization strategies, which lack theoretical support. (2) The advantage of our method becomes more pronounced, indicating that our method enhances the ability of pruned models even at extremely high sparsity levels.

In Table 1, we further observe that SparseGPT, which updates the values of the remaining weights, outperforms Wanda, which leaves the remaining weights unchanged. As sparsity increases, the advantage of SparseGPT over Wanda becomes more evident, particularly at high sparsity levels. Additionally, the DsnoT approach, which reselects the masks after pruning and keep weights unchanged, also faces challenges. For example, when the sparsity exceeds 70%, regardless of LlamaV1 or LlamaV2, DsnoT significantly decreases the performance of the sparse model pruned by SparseGPT. In contrast, our method effectively and efficiently fine-tunes the weights of the LLM block by block, surpassing other baselines overall. In the later section, we will conduct comprehensive and detailed experiments to further compare mask-tuning and weight-tuning.

Method \ Sparsity	LlamaV1-7B 2 : 4	4 : 8	LlamaV2-7B 2 : 4	4 : 8
Magnitude	42.54	16.83	54.39	16.53
w. DsnoT	38.32	17.01	40.81	18.34
w. Ours	**9.62**	**8.10**	**9.14**	**7.56**
Wanda	11.50	8.57	12.11	8.66
w. DsnoT	10.95	8.46	11.98	8.57
w. Ours	**8.89**	**7.66**	**8.30**	**7.11**
SparseGPT	11.05	8.55	10.44	8.01
w. DsnoT	10.00	8.26	10.06	8.06
w. Ours	**8.82**	**7.59**	**8.25**	**7.06**

Table 2: Comparison of perplexity for pruning and fine-tuning LlamaV1-7B and LlamaV2-7B on the Wikitext2 dataset at N:M sparsity levels, including two patterns, 2:4 and 4:8.

Semi-structured Pruning. Semi-structured pruning, also known as N:M sparsity, is considered superior to unstructured pruning when it comes to accelerating models on devices. We conducted extensive comparison experiments on the Wikitext2 dataset, and the results are presented in Table 2. Irrespective of the 2:4 or 4:8 pattern, our method consistently outperforms DsnoT, significantly enhancing the performance of the pruned models. For example, when using the 2:4 pattern and Wanda mask initialization, our method achieves a perplexity of 8.30 for the sparse LlamaV2 model, which even surpasses the performance of DsnoT using the 4:8 pattern. The sparse LLMs pruned by magnitude pruning and fine-tuned with our method demonstrate a remarkable improvement. Our fine-tuning

approach can effectively narrow the performance gap between magnitude pruning and the state-of-the-art baselines, Wanda and SparseGPT.

4.2 Zero-Shot Tasks

We conducted extensive experiments to evaluate the performance of the sparse model on 7 zero-shot tasks. The metric we used is accuracy. The experimental results of different methods at the unstructured sparsity level are shown in Table 3. It can be observed that EBFT significantly enhances the generality of the pruned model. For instance, with magnitude pruning, EBFT improves the accuracy by 16.28 on LlamaV1-7B and by 13.53 on LlamaV2-7B. With Wanda, our methods achieve a mean accuracy of 61.14 on LlamaV1-7B and 61.12 on LlamaV2-7B. However, DSnoT hardly enhances the performance of the pruned model. It achieves a mean accuracy of 58.85 on LlamaV1-7B and 58.25 on LlamaV2-7B, respectively. For LlamaV2, DSnoT even degrades the performance after fine-tuning. The mean accuracy before fine-tuning for Wanda and SparseGPT is 59.02 and 60.77, respectively. After fine-tuning, the mean accuracy drops to 58.25 for Wanda and 60.20 for SparseGPT, highlighting the limitations of DSnoT. In contrast, after fine-tuning with EBFT, the sparse LlamaV2 model shows a significant improvement in overall accuracy, with a mean accuracy of 61.12 for Wanda and 61.96 for SparseGPT.

N:M sparsity. We also investigated the generality of our EBFT approach at N:M sparsity levels. Similar to unstructured sparsity, EBFT demonstrates significant advantages compared to other baselines. The experimental results for the 2:4 pattern are presented in Tab.3. In the case of magnitude pruning, EBFT improves the mean accuracy of sparse LlamaV1-7B by 6.39 and sparse LlamaV2-7B by 2.8. Conversely, DSnoT fails to restore the performance of magnitude-pruned sparse models. When using Wanda initialization, EBFT enhances the mean accuracy of sparse LlamaV1-7B by 2.11 and sparse LlamaV2-7B by 2.33. Under SparseGPT initialization, EBFT improves the mean accuracy of sparse LlamaV1-7B by 1.93 and sparse LlamaV2-7B by 1.38. In contrast, DSnoT loses its effectiveness with the current pattern as a fine-tuning method. Excluding the SparseGPT initialization on LlamaV2-7B, DSnoT significantly degrades the accuracy of the sparse model. For instance, with Wanda initialization, it results in a drop of 2.85 in accuracy for LlamaV1-7B and 0.42 for LlamaV2-7B.

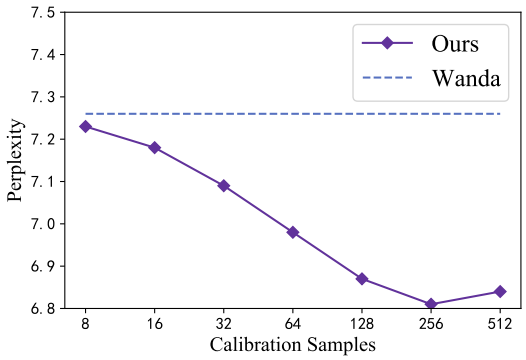

Figure 2: The perplexity of the fine-tuned LlamaV1-7B on Wikitext2, with a sparsity level of 50%, varies with the number of samples in the calibration dataset.

4.3 Calibration Samples

We vary the number of samples in the calibration dataset and generate a plot illustrating the perplexity and number of samples for the fine-tuned sparse LlamaV1-7B under Wanda initialization. The results are presented in Fig.2. The experimental findings demonstrate the robustness of our proposed method. Generally, as the number of samples increases, the performance of the sparse model improves. However, once the number of samples reaches 512, the perplexity does not decrease further. Notably, even with just 8 samples, the fine-tuned sparse LlamaV1 model exhibits an improvement compared to the model before fine-tuning. In addition, as the number of samples decreases, the fine-tuning speed can be further accelerated.

4.4 EBFT vs. LoRA

Low-Rank Adaptation (LoRA) has gained popularity as a technique for retraining large language models. Recent works such as (Ma et al., 2023; Guo et al., 2023a; Li et al., 2023) have extensively used LoRA for retraining pruned models. This involves fine-tuning the low-rank parameters A and B of an additional adapter on a large dataset to restore its performance. In this paper, we provide a detailed comparison of the fine-tuning cost and performance between LoRA and our EBFT.

In our study, we applied Low-Rank Adaptation (LoRA) and EBFT to FLAP (An et al., 2023) with structured sparsity levels. FLAP is a state-of-the-art method that outperforms LLM-Pruner in various tasks. It introduces a novel metric for channels in large language models and utilizes this metric score to search for the global structure of the model. We

Model	Method	PIQA	ARC-E	ARC-C	WinoGrande	HellaSwag	Boolq	StoryCloze	Mean
Lla.1(60%)	Mag.	60.55	42.30	23.21	50.04	31.86	38.29	57.40	43.38
	w.DSnoT	66.65	51.01	26.02	52.96	38.31	46.82	65.37	49.59
	w.Ours	**72.69**	**63.26**	**32.17**	**63.85**	**46.61**	**65.72**	**73.33**	**59.66**
	Wanda	72.74	62.67	30.03	62.67	43.71	68.90	71.25	58.85
	w.DSnoT	73.07	63.38	30.80	61.56	43.51	68.20	71.46	58.85
	w.Ours	**73.67**	**65.57**	**32.17**	**65.11**	**47.80**	**69.79**	**73.86**	**61.14**
	SparseGPT	72.36	62.58	31.14	64.40	45.38	**69.79**	73.65	59.90
	w.DSnoT	73.70	63.17	31.83	63.06	47.41	67.52	73.22	59.99
	w.Ours	**73.77**	**64.02**	**32.51**	**64.40**	**47.84**	69.27	**74.02**	**60.83**
Lla.2(60%)	Mag.	62.73	44.78	25.00	53.12	34.99	47.86	62.21	47.24
	w.DSnoT	69.42	63.13	30.89	61.56	40.48	54.77	67.93	55.46
	w.Ours	**72.63**	**64.94**	**32.25**	**65.11**	**46.40**	**71.01**	**73.06**	**60.77**
	Wanda	71.71	64.98	30.55	**64.56**	43.82	65.57	71.99	59.02
	w.DSnoT	71.33	64.44	29.95	64.17	42.53	64.83	70.55	58.25
	w.Ours	**73.56**	**68.73**	**33.19**	64.40	**47.26**	**67.22**	**73.49**	**61.12**
	SparseGPT	71.44	63.72	31.48	66.69	45.25	72.54	**74.29**	60.77
	w.DSnoT	72.85	66.58	33.19	62.83	46.71	65.72	73.54	60.20
	w.Ours	**73.29**	**67.42**	**32.59**	**66.98**	**47.10**	**72.60**	73.70	**61.96**
Lla.1(2: 4)	Mag.	68.01	53.32	27.22	59.91	42.30	53.09	70.02	53.41
	w.DSnoT	68.18	54.38	26.54	58.96	41.24	48.32	68.68	52.33
	w.Ours	**72.80**	**64.18**	**30.89**	**64.25**	**45.80**	**68.29**	**72.37**	**59.80**
	Wanda	70.40	60.82	27.79	63.22	42.08	69.08	70.71	57.76
	w.DSnoT	70.62	61.78	28.07	61.56	42.35	48.32	70.71	54.91
	w.Ours	**72.42**	**64.81**	**30.97**	**65.19**	**46.05**	**67.25**	**72.42**	**59.87**
	SparseGPT	71.22	60.73	30.46	63.38	42.95	**69.85**	70.23	58.40
	w.DSnoT	72.63	63.13	30.72	62.67	45.91	67.77	71.73	59.22
	w.Ours	**73.45**	**64.77**	**30.80**	**66.30**	**46.39**	68.44	**72.15**	**60.33**
Lla.2(2: 4)	Mag.	70.08	61.91	30.12	60.93	**45.43**	59.85	72.31	57.23
	w.DSnoT	69.10	61.45	29.01	59.12	43.75	65.37	70.82	55.76
	w.Ours	**73.07**	**67.17**	**30.72**	**64.64**	45.27	**66.73**	**72.63**	**60.03**
	Wanda	70.89	61.91	30.72	62.51	41.27	68.53	70.23	58.01
	w.DSnoT	70.18	61.74	29.78	62.75	40.90	67.86	69.91	57.59
	w.Ours	**72.91**	**65.91**	**31.91**	**63.77**	**45.49**	**69.33**	**73.06**	**60.34**
	SparseGPT	70.40	63.80	31.23	65.75	43.83	68.04	73.06	59.44
	w.DSnoT	73.34	65.24	**32.17**	63.14	45.41	67.55	73.76	60.09
	w.Ours	**73.34**	**66.33**	30.80	**65.88**	**45.80**	**69.79**	**73.76**	**60.82**

Table 3: Accuracy results of pruning and fine-tuning LlamaV1-7B and LlamaV2-7B on a series of zero-shot tasks at **60%** sparsity and **2:4** pattern sparsity.

utilized the masks generated by FLAP as initialization for the fine-tuning process.

When fine-tuning the model pruned by FLAP using LoRA, we selected the Alpaca-GPT4 dataset as the retraining dataset. The Alpaca-GPT4 dataset consists of 50k rows of data and was fine-tuned using GPT4. We performed fine-tuning with LoRA for 2 epochs on the Alpaca-GPT4 dataset, using a learning rate of 1e-4 and a batch size of 64, which is the same as LLM-Pruner.

The fine-tuning methods employed in recent state-of-the-art works, as mentioned above, can incur a significant retraining cost. We compared their retraining methods with ours on a 40G A100 GPU. The time costs and perplexity on Wikitext2 of LoRA and EBFT are listed in Table 4. It is observed that compared to LoRA, our EBFT achieves a 10× speedup, resulting in a significant reduction in fine-tuning costs. Additionally, EBFT demonstrates better performance compared to LoRA. As shown in Table 4, when reducing 20% of the parameters of LlamaV2-7B, EBFT achieves a perplexity of 15.71 on Wikitext2, which is superior to the perplexity obtained by LoRA (16.08).

Method	sparsity	time	perplexity
LoRA	20%	5h	16.08
Ours	20%	**0.5h**	**15.71**

Table 4: The time cost and perplexity of LoRA and EBFT on the LlamaV2-7B at sparsity of 20%.

We further conducted detailed experiments to compare our method with LoRA. We varied the parameters of the pruned models, including LlamaV1-7B and LlamaV2-7B, and evaluated the perplexity and accuracy of the fine-tuned models on Wikitext2

Model	Param.	Method	ARC-E	ARC-C	PIQA	WinoGrande	StoryCloze	Boolq	Mean	wiki.ppl
	5.5B	LoRA	64.31	37.46	**76.66**	64.64	**77.28**	**71.47**	65.30	15.46
	5.5B	**Ours**	**72.52**	**38.65**	75.46	**66.46**	75.63	71.19	**66.65**	**14.81**
Lla.1	5.0B	LoRA	60.48	33.87	**75.08**	61.80	**76.16**	63.82	61.87	16.67
	5.0B	**Ours**	**68.31**	**33.96**	72.85	**63.85**	73.45	**68.90**	**63.55**	**16.27**
	5.5B	LoRA	64.35	34.90	**75.84**	62.51	**75.47**	50.06	60.52	16.08
	5.5B	**Ours**	**68.81**	**35.24**	74.81	**63.93**	72.31	**60.73**	**62.64**	**15.71**
Lla.2	5.0B	LoRA	61.32	32.08	**73.78**	61.96	**74.13**	55.05	59.72	**17.63**
	5.0B	**Ours**	**65.74**	**32.76**	71.87	**64.40**	71.04	**59.39**	**60.87**	**17.63**

Table 5: The accuracy and perplexity of the fine-tuned LlamaV1-7B and LlamaV2-7B models on Wikitext2, as well as their performance on a series of zero-shot tasks. The pruned models used in our experiments have parameters set at 5.5B and 5B, respectively.

as well as a series of zero-shot tasks. The experimental results are summarized in Tab.5. Indeed, the comparison between EBFT and LoRA continues to demonstrate the advantages of EBFT. For example, after fine-tuning LlamaV1-5.5B, EBFT achieves a perplexity of 14.81, surpassing LoRA, which achieves a perplexity of 15.46 on Wikitext2. Similarly, for LlamaV2-5.5B, EBFT achieves a perplexity of 15.71, outperforming LoRA with a perplexity of 16.08. This trend carries over to the zero-shot tasks as well, where the fine-tuned models using EBFT exhibit better performance compared to LoRA. The mean accuracy of our approach is higher than that of LoRA, regardless of whether it is applied to LlamaV1 or LlamaV2. While it is true that LoRA may achieve better scores on certain tasks such as PIQA and StoryCloze, the overall results consistently support the conclusion that the pruned models fine-tuned using EBFT outperform those fine-tuned using LoRA. **When comparing EBFT to LoRA**, EBFT demonstrates **faster speed**, **lower cost**, and **superior performance**.

4.5 Weight Tuning vs. Mask Tuning

Some optimization methods for sparse models, such as (Zhang et al., 2023b,d), solely update the positions of masks without adjusting weights. To explore the effectiveness of this strategy, we conducted experiments to compare two fine-tuning strategies: weight tuning and mask tuning.

For mask tuning, we employed Eq.4 as the optimization objective, aiming to minimize the block-wise reconstruction error. The fine-tuning process of mask tuning is the same as that of EBFT, except that mask tuning only updates the positions of masks while keeping the weights unchanged. We recorded the experimental results in Tab.6. Specifically, we conducted variations in the sparsity levels of LlamaV1-7B and LlamaV2-7B, and evaluated the perplexity of the fine-tuned sparse models on Wikitext2. The results consistently highlight the clear advantage of weight tuning over mask tuning, even though the mask tuning method used in this study outperforms the SOTA mask-tuning method DSnoT in Tab.1. However, mask tuning still falls short when compared to EBFT. Regardless of the sparsity level, weight tuning consistently outperforms mask tuning. These findings clearly indicate the limitations of mask-tuning methods.

LlamaV1-7B					
Method	50%	60%	70%	80%	90%
w.Mask	7.05	9.15	25.90	456.0	5378
w.Weight	**6.81**	**8.59**	**16.88**	**118.4**	**2993**
LlamaV2-7B					
Method	50%	60%	70%	80%	90%
w.Mask	6.29	8.40	26.99	755.8	3793
w.Weight	**6.18**	**7.90**	**16.94**	**72.80**	**903.4**

Table 6: The Wikitext2 perplexity of mask-tuning and weight-tuning were evaluated on LlamaV1-7B and LlamaV2-7B at various sparsity levels with Wanda initialization.

5 Conclusion

We propose EBFT, a unified fine-tuning framework for sparse Language Models that can be integrated with any pruning method. In EBFT, we define the block-wise reconstruction error and optimize it on a block-by-block basis through backpropagation algorithm, aiming to achieve a convergent and optimal solution. This approach proves to be effective and efficient, requiring only a small number of samples for calibration. Extensive experiments demonstrate that EBFT achieves state-of-the-art performance on various benchmark datasets.

6 Limitation

Although the use of a small calibration dataset significantly reduces costs, the fine-tuning process of EBFT still incurs computation costs due to gradient calculations. In future work, we will continue to focus on fine-tuning with a limited number of samples and explore gradient-free methods to further mitigate these costs.

References

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. 2023. Fluctuation-based adaptive structured pruning for large language models. *arXiv preprint arXiv:2312.11983*.

Saleh Ashkboos, Maximilian L Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. 2024. Slicegpt: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024*.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Vladimír Boža. 2024. Fast and optimal weight update for pruned large language models. *arXiv preprint arXiv:2401.02938*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Together Computer. 2023. Redpajama: an open dataset for training large language models.

Rocktim Jyoti Das, Liqun Ma, and Zhiqiang Shen. 2023. Beyond size: How gradients shape pruning decisions in large language models. *arXiv preprint arXiv:2311.04902*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*.

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

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*.

Song Guo, Jiahang Xu, Li Lyna Zhang, and Mao Yang. 2023a. Compresso: Structured pruning with collaborative prompting learns compact large language models. *arXiv preprint arXiv:2310.05015*.

Song Guo, Lei Zhang, Xiawu Zheng, Yan Wang, Yuchao Li, Fei Chao, Chenglin Wu, Shengchuan Zhang, and Rongrong Ji. 2023b. Automatic network pruning via hilbert-schmidt independence criterion lasso under information bottleneck principle. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 17458–17469.

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.

Babak Hassibi, David G Stork, and Gregory J Wolff. 1993. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, pages 293–299. IEEE.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. *arXiv preprint arXiv:2305.02301*.

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

Itay Hubara, Brian Chmiel, Moshe Island, Ron Banner, Joseph Naor, and Daniel Soudry. 2021. Accelerated sparse neural training: A provable and efficient method to find n: m transposable masks. *Advances in neural information processing systems*, 34:21099–21111.

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.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. 2016. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*.

Yixiao Li, Yifan Yu, Qingru Zhang, Chen Liang, Pengcheng He, Weizhu Chen, and Tuo Zhao. 2023. Losparse: Structured compression of large language models based on low-rank and sparse approximation. *arXiv preprint arXiv:2306.11222*.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. 2023. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*.

Christos Louizos, Max Welling, and Diederik P Kingma. 2017. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. *arXiv preprint arXiv:2305.11627*.

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

Nasrin Mostafazadeh, Michael Roth, Annie Louis, Nathanael Chambers, and James Allen. 2017. Lsdsem 2017 shared task: The story cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 46–51.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

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

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*.

Aaquib Syed, Phillip Huang Guo, and Vijaykaarti Sundarapandiyan. 2023. Prune and tune: Improving efficient pruning techniques for massive language models.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Stanford alpaca: an instruction-following llama model (2023). *URL https://github. com/tatsu-lab/stanford_alpaca*.

Naftali Tishby, Fernando C Pereira, and William Bialek. 2000. The information bottleneck method. *arXiv preprint physics/0004057*.

Naftali Tishby and Noga Zaslavsky. 2015. Deep learning and the information bottleneck principle. In *2015 ieee information theory workshop (itw)*, pages 1–5. IEEE.

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. 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.

Ziheng Wang, Jeremy Wohlwend, and Tao Lei. 2019. Structured pruning of large language models. *arXiv preprint arXiv:1910.04732*.

Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2023. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694*.

Mengzhou Xia, Zexuan Zhong, and Danqi Chen. 2022. Structured pruning learns compact and accurate models. *arXiv preprint arXiv:2204.00408*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Mingyang Zhang, Chunhua Shen, Zhen Yang, Linlin Ou, Xinyi Yu, Bohan Zhuang, et al. 2023a. Pruning meets low-rank parameter-efficient fine-tuning. *arXiv preprint arXiv:2305.18403*.

Yuxin Zhang, Mingbao Lin, Zhihang Lin, Yiting Luo, Ke Li, Fei Chao, Yongjian Wu, and Rongrong Ji. 2022. Learning best combination for efficient n: M sparsity. *Advances in Neural Information Processing Systems*, 35:941–953.

Yuxin Zhang, Mingbao Lin, Yunshan Zhong, Fei Chao, and Rongrong Ji. 2023b. Lottery jackpots exist in pre-trained models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Yuxin Zhang, Yiting Luo, Mingbao Lin, Yunshan Zhong, Jingjing Xie, Fei Chao, and Rongrong Ji. 2023c. Bi-directional masks for efficient n: M sparse training. *arXiv preprint arXiv:2302.06058*.

Yuxin Zhang, Lirui Zhao, Mingbao Lin, Yunyun Sun, Yiwu Yao, Xingjia Han, Jared Tanner, Shiwei Liu, and Rongrong Ji. 2023d. Dynamic sparse no training: Training-free fine-tuning for sparse llms. *arXiv preprint arXiv:2310.08915*.

Aojun Zhou, Yukun Ma, Junnan Zhu, Jianbo Liu, Zhijie Zhang, Kun Yuan, Wenxiu Sun, and Hongsheng Li. 2021. Learning n: m fine-grained structured sparse neural networks from scratch. *arXiv preprint arXiv:2102.04010*.