MINI-LLM: Memory-Efficient Structured Pruning for Large Language Models

Hongrong Cheng[1] , **Miao Zhang**[2*] , **Javen Qinfeng Shi**[1]

[1]University of Adelaide, Adelaide, Australia
[2]Harbin Institute of Technology, Shenzhen, China

{hongrong.cheng, javen.shi}@adelaide.edu.au, zhangmiao@hit.edu.cn

Abstract

As Large Language Models (LLMs) grow dramatically in size, there is an increasing trend in compressing and speeding up these models. Previous studies have highlighted the usefulness of gradients for importance scoring in neural network compressing, especially in pruning medium-size networks. However, the substantial memory requirements involved in calculating gradients with backpropagation impede the utilization of gradients in guiding LLM pruning. As a result, most pruning strategies for LLMs rely on gradient-free criteria, such as weight magnitudes or a mix of magnitudes and activations. In this paper, we devise a hybrid pruning criterion, which appropriately integrates magnitude, activation, and gradient to capitalize on feature map sensitivity for pruning LLMs. To overcome memory requirement barriers, we estimate gradients using only forward passes. Based on this, we propose a Memory-effIcieNt structured prunIng procedure for LLMs (MINI-LLM) to remove no-critical channels and multi-attention heads. Experimental results demonstrate the superior performance of MINI-LLM over existing gradient-free methods on three LLMs: LLaMA, BLOOM, and OPT across various downstream tasks (classification, multiple-choice, and generation), while MINI-LLM maintains a GPU memory footprint akin to gradient-free methods.

1 Introduction

The advent of pre-trained Large Language Models (LLMs), such as GPT-4 [OpenAI, 2023] and LLaMA [Touvron *et al.*, 2023], has made remarkable processes across various complex Natural Language Processing (NLP) tasks, such as natural language generation [Wu *et al.*, 2020], question answering [Brown *et al.*, 2020], and recommendation system [Wu *et al.*, 2023]. However, this remarkable capability usually entails a large model size, resulting in significant computational costs in terms of storage, memory, and computation time, which presents considerable difficulties during the training and deployment phases. To this end, there has been considerable interest in compressing LLMs [Ma *et al.*, 2023; Dettmers *et al.*, 2023; Frantar and Alistarh, 2023; Xiao *et al.*, 2023; Li *et al.*, 2020] to make them more practical for various tasks. Neural network pruning [Ma *et al.*, 2023; Frantar and Alistarh, 2023; Sun *et al.*, 2024; Xia *et al.*, 2024], as one of the indispensable approaches for compressing and accelerating neural networks, has recently found its way into LLMs.

In the traditional pruning methods [Molchanov *et al.*, 2017; Lee *et al.*, 2019; Sanh *et al.*, 2020; Liu *et al.*, 2021; Fu *et al.*, 2022] for compressing small or medium-size models, gradients of loss functions w.r.t. weights, masks, or feature maps have demonstrated more reliable performance than gradient-free methods (e.g., magnitude-based methods) in discriminating important weights/channels. For example, [Lee *et al.*, 2019] exploits the first-order Taylor expansion to identify the connection sensitivity caused by setting some weights to zero, which outperforms magnitude-based methods and obtains extremely sparse networks with similar accuracy as the reference networks. However, due to the huge number of parameters in LLMs, computing gradients with backpropagation requires a prohibitive amount of memory. LLM-Pruner [Ma *et al.*, 2023], which uses gradients calculated via backpropagation for structured pruning, consumes about twice the GPU resources compared to magnitude-based methods during pruning LLaMA-7B, as illustrated in Figure 1[1]. Even though the recovery stage can be executed on a single RTX 4090 (24GB) with the help of Low-Rank Adaption (LoRA) [Hu *et al.*, 2022], the GPU consumption during the pruning stage has become the bottleneck for GPU resource usage in the entire pruning framework.

To avoid incurring untenable memory costs for computing gradients on LLMs, some pruning methods [Frantar and Alistarh, 2023; Sun *et al.*, 2024] constructed gradient-free criteria. For example, SparseGPT [Frantar and Alistarh, 2023] employed the combination of weight magnitude and the inverse Hessian matrix which is formed from the product of given input features to score weights for unstructured pruning. However, the computation of the inverse Hessian matrix is resource-intensive, and the utility of gradient information remains under-exploited. Some pruning methods, such as

---

*Corresponding author.

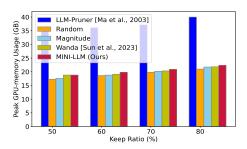[1]We obtained the data on an NVIDIA A100 (40GB)

Figure 1: The peak GPU-memory Usage for pruning LLaMA-7B. The backpropagation gradient-based pruning method, LLM-Pruner, consumes about twice the GPU resources compared to gradient-free methods and our method MINI-LLM during pruning LLaMA-7B.

Sheared LLaMA [[Xia *et al.*, 2024]], combine pruning with pre-training, which requires substantial GPU resources. For example, Sheared LLaMA requires 8 A100 (80GB) GPUs for pruning and 16 for pre-training. Since this paper focuses on memory-efficient pruning methods and follows the mainstream framework that starts with pruning and then fine-tuning, without pre-training, although outstanding in performance, these methods are not within the scope of our discussion.

In this paper, we propose the Memory-effIcieNt structured prunIng procedure for LLMs (MINI-LLM), which scores weights using estimated gradients with only forward passes. We make this approach tractable by contributing multiple techniques.

Our main contributions are as follows:

- We design a novel pruning criterion called Feature Map Sensitivity (FMS) score, integrating weight magnitude, activation, and gradient. This criterion optimally utilizes the pivotal information from the three critical aspects, which facilitates a more nuanced assessment of feature map sensitivity and provides effective scoring in LLMs.

- We propose a structured pruning framework for LLMs called MINI-LLM which utilizes estimated gradients with only forward passes by using comparable GPU memory usage to gradient-free methods, significantly improving GPU memory efficiency over traditional backpropagation gradients.

- The experiments on three types of LLMs: LLaMA, BLOOM, and OPT over different downstream tasks (classification, multiple-choices, and generation) demonstrate that the novel pruning criterion FMS can effectively boost the performance of gradient-based methods. Additionally, our proposed gradient-based structured pruning method MINI-LLM steadily exceeds gradient-free pruning methods in performance and rivals or surpasses backpropagation gradient-based method at times, while using similar GPU memory as gradient-free methods.

2 Related Work

Structured/Unstructured/Semi-structured LLM pruning. The pruning methods for LLMs can still be generally cat-

egorized as unstructured ([Sun *et al.*, 2024; Frantar *et al.*, 2022]), semi-structured ([Frantar and Alistarh, 2023]), and structured ([Ma *et al.*, 2023; Wang *et al.*, 2020b]) pruning methods, similar to the categorization for pruning small and mid-size neural networks. Unstructured pruning achieves substantial sparsity by directly setting weights or their masks to zero while maintaining a comparable performance compared to the vanilla models. However, the irregular sparsity results in no compression in the model size, and actual acceleration necessitates the support of specialized software/hardware. In contrast, structured pruning discards the whole grouped parameters (such as channels and attention heads), leading to physically reduced model size and enabling inference acceleration without any special requirements of software/hardware ([Zhou *et al.*, 2022; Frantar and Alistarh, 2023]). Semi-structure pruning, such as 2:4 or 4:8 patterns in [Frantar and Alistarh, 2023], provides a balance between performance and hardware speedup. In this paper, we focus on structured pruning for LLMs.

Pruning criteria for LLMs. Neural network pruning methods search for an optimal subnetwork by removing unimportant weights. As one of the most popular criterion factors, gradients have already been demonstrated effective in constructing scoring functions for pruning small or medium-size networks [Liu *et al.*, 2021; Fu *et al.*, 2022; Wang *et al.*, 2020a; Yu *et al.*, 2022; Molchanov *et al.*, 2019; Kwon *et al.*, 2022]. However, calculating gradients using backpropagation is highly resource-intensive for GPU memory, making it challenging to implement for LLMs, where meeting such high memory demands is difficult. For example, LLM-Pruner [Ma *et al.*, 2023] employs gradients calculated through backpropagation for LLM pruning, but the GPU memory required for pruning exceeds that of fine-tuning. To this end, there are some gradient-free pruning methods ([Frantar and Alistarh, 2023; Sun *et al.*, 2024; Nova *et al.*, 2023; Kurtic *et al.*, 2023; Li *et al.*, 2022b]). Most of them are centered on post-training (retraining-free) approaches that involve pruning while concurrently compensating for performance. For instance, Wanda [Sun *et al.*, 2024] multiplies weight magnitude and the corresponding activation to implement unstructured post-training pruning for LLMs. Even though these gradient-free methods are GPU memory efficient, the utility of gradient remains under-exploited. This paper actively seeks an effective estimation method to overcome memory requirement barriers related to computing gradients with backpropagation.

Zeroth-Order optimization. Zeroth-Order (ZO) optimization can fall in the general class of weight perturbation methods. An early method referred to as the Finite Difference Stochastic Approximation (FDSA) [Kiefer and Wolfowitz., 1952] estimated the gradient by using $2d$ function measurements, two for each of the d partial derivatives. One of the key disadvantages of FDSA is that a large d value would result in serious computational challenges. A more efficient gradient estimation method is Simultaneous Perturbation Stochastic Approximation (SPSA) [Spall, 1992; Li *et al.*, 2022a] which approximates gradients using only two forward passes. In previous work, ZO gradient estima-

tion was used for solving optimization-related problems, such as for model training. For example, Malladi et al. [Malladi *et al.*, 2023] propose a Memory-efficient ZO-SGD (MeZO) to adapt SPSA to fine-tuning LLMs in a memory-efficient way. In this paper, for the first time, we apply ZO gradient estimation for pruning LLMs.

3 Method

In this section, we propose a Memory-effIcieNt structured prunIng procedure for LLMs termed MINI-LLM. We start by describing a new pruning criterion that evaluates feature map saliency from three critical factors: gradient, weight magnitude, and activation. To evaluate gradients in a memory-efficient way, we exploit ZO gradients to approximate the backpropagation based gradients. Finally, to recover performance, we utilize LoRA [Hu *et al.*, 2022] to fine-tune the pruned model, which has high training throughput, but low GPU memory requirement.

3.1 Pruning Criterion in MINI-LLM

Problem Definition. The pruning problem for LLMs starts from a pre-trained dense model $W_0 \in \mathbb{R}^d$ and aims to find a sparse version of W_0, where many channels and attention heads are discarded. The remaining weights $\hat{W}_0$ may be updated accordingly to preserve the performance. Consider a labeled dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, where N is the number of samples, and a desired prune ratio p (i.e., the percentage of removed weights), our goal is to remove the weights that has the least impact on the model's prediction. Therefore, neural network pruning can be formulated as the following constrained optimization problem:

$$\min_{\hat{W}_0} \mathcal{L}(\hat{W}_0; \mathcal{D}) = \min_{\hat{W}_0} \frac{1}{N} \sum_{i=1}^N \ell(\hat{W}_0; (x_i, y_i)), \quad (1)$$
$$\text{s.t.} \quad ||\hat{W}_0||_0 \leq ||W_0||_0 \times (1-p),$$

where $\ell(\cdot)$ can be the standard loss function (e.g., cross-entropy loss) and $||\cdot||_0$ is the standard L_0 norm.

Gradient-based Pruning. To evaluate the significance of a specific weight W_l^k, one common way is using its sensitive effect on the loss function, where k denotes the k-th weight in the l-th layer. Specifically, one can compare the difference in the loss function when W_l^k is included versus when it is excluded from the model (i.e., LLaMA-7B). Thus, the loss change can be formulated as [LeCun *et al.*, 1989]:

$$\Delta \mathcal{L} = \mathcal{L}_{W_l^k}(\mathcal{D}) - \mathcal{L}_{W_l^k=0}(\mathcal{D})$$
$$\approx \Delta W^T \frac{\partial \mathcal{L}(\mathcal{D})}{\partial W_l^k} - \frac{1}{2} \Delta W^T H \Delta W \quad (2)$$
$$= W_l^k \frac{\partial \mathcal{L}(\mathcal{D})}{\partial W_l^k} - \frac{1}{2} W_l^k H W_l^k,$$

where $\Delta W = W_l^k$ and the Hessian matrix $H = \nabla_{W_l^k}^2 \mathcal{L}(W_0)$.

Unlike previous work [Liu *et al.*, 2021; Kurtic *et al.*, 2022], the pre-trained datasets of a large language model are inconsistent with the datasets of the downstream tasks, hence

$\frac{\partial \mathcal{L}(\mathcal{D})}{\partial W_l^k} \not\approx 0$. This characteristic is advantageous for assessing the importance of weights through the gradient term in the context of LLMs, as calculating the Hessian matrix in the second term is impractical on LLMs with $\mathcal{O}(N^2)$ complexity. Therefore, $\Delta \mathcal{L}$ can be approximated as:

$$\Delta \mathcal{L} = \mathcal{L}_{W_l^k}(\mathcal{D}) - \mathcal{L}_{W_l^k=0}(\mathcal{D}) \approx W_l^k \frac{\partial \mathcal{L}(\mathcal{D})}{\partial W_l^k}. \quad (3)$$

Activation-based Pruning. As recently observed in LLMs larger than 6.7B ([Dettmers *et al.*, 2022]), a small set of hidden state features emerges with significantly larger magnitudes (outliers) than the remainders and zeroing out these features causes a significant degradation of performance. The vanilla scoring function Eq. (3) does not highlight the unique characteristics of LLMs compared with smaller models. Given the lth layer's input activation X_l (i.e., the output from the $(l-1)$th layer of the network) and the lth layer's weights W_l, the pruning problem is also commonly treated as finding the subset $\hat{W}_l$ of W_l respecting a compression constraint $\mathcal{C}$, which most closely approximates the initial output as determined by the squared error metric. Assuming that the activation and weight matrices possess a suitable rectangular shape, the neural network pruning is defined as the following optimization problem [Frantar and Alistarh, 2023]:

$$\min_{\hat{W}_l} ||\hat{W}_l X_l - W_l X_l||_2^2 = ||\Delta W_l X_l||_2^2, \quad (4)$$
$$\text{s.t.} \quad \hat{W}_l \in \mathcal{C}.$$

To evaluate the significance of a specific weight W_l^k, one can compare the difference in the layer-wise output when W_l^k is preserved versus when it is excluded from the model and write the formulation as:

$$||\Delta W_l X_l||_2 = |W_l^k X_l^k|, \quad (5)$$

where $\Delta W_l = W_l^k$.

Feature Map Sensitivity (FMS). Eq. (5) only considers the output changes in a single layer and does not take into account the global loss change across the entire network. We notice that, with the weight gradients, the global loss change can be quantified with weight change as shown in Eq. (3). To calculate the salience of each weight relative to the change in global loss and layer-wise output, we measure ΔW_l in Eq. (5) through its global sensitivity $\Delta \mathcal{L}$ from Eq. (3) and propose a heuristic hybrid sensitivity scoring function called Feature Map Sensitivity (**FMS**) as follows:

$$S(W_l^k)_{ours} = \left| W_l^k \frac{\partial \mathcal{L}(\mathcal{D})}{\partial W_l^k} X_l^k \right|. \quad (6)$$

Compared to Eq. (3) and Eq. (5), our criterion Eq. (6) integrates magnitude, activation, and gradient to optimally utilize the pivotal information from the three critical aspects, so as calculate the feature map sensitivity along with the loss changes.

3.2 Pruning with Estimated Gradients

Let $\mathcal{L}(W; \mathcal{B})$ denote the loss on a minibatch $\mathcal{B} \subset \mathcal{D}$. The following Definition 1 describes a classical ZO gradient estimation based on SPSA ([Spall, 1992]).

Algorithm 1 The structured pruning algorithm MINI-LLM

Input: Dataset $\mathcal{D}$, pre-trained weights $W_0 \in \mathbb{R}^d$, loss $\mathcal{L}$: $\mathbb{R}^d \to \mathbb{R}$, prune ratio p, perturbation scale ϵ.
Output: The pruned model

1: Clear every weight's sensitivity score $\hat{S}(W_l^k) = 0$;
2: Forward via Eq. (7) and estimate each weight's $\hat{g}_l^k$;
3: **for** $l \in [1, ..., L]$ **do**
4:     Compute the input activation X_l for l-th layer;
5:     Compute every weight's score $\hat{S}(W_l^k)$ via Eq. (9);
6: **end for**
7: **for** $l \in [1, ..., L]$ **do**
8:     Keep the important groups $\hat{S}(G_l)$ ranked in top $1 - p$;
9: **end for**
10: **return** the pruned model.

Definition 1 (ZO Gradient Estimation.). *Given a model with parameters $W \in \mathbb{R}^d$ and a loss function $\mathcal{L}$, ZO gradient on a minibatch $\mathcal{B}$ is as*

$$\hat{\nabla}\mathcal{L}(W; \mathcal{B}) = \frac{\mathcal{L}(W+\epsilon z; \mathcal{B}) - \mathcal{L}(W-\epsilon z; \mathcal{B})}{2\epsilon z} \approx \nabla\mathcal{L}(W; \mathcal{B}), \quad (7)$$

where $\nabla\mathcal{L}(W; \mathcal{B})$ is the gradient with backpropagation, $z \in \mathbb{R}^d$ with $z \sim \mathcal{N}(0, I_d)$ and ϵ is the perturbation scale. The n-ZO gradient estimate averages $\hat{\nabla}\mathcal{L}(W; \mathcal{B})$ over n randomly sampled z. Malladi et al. [2023] found that $n = 1$ is the most efficient. Therefore, we choose $n = 1$ as the default. For each weight W_l^k in the model, the estimation of its gradient $\frac{\partial \mathcal{L}(\mathcal{D})}{\partial W_l^k}$ (defined as $\hat{g}_l^k$) is then

$$\hat{g}_l^k = \frac{\mathcal{L}(W+\epsilon z; \mathcal{B}) - \mathcal{L}(W-\epsilon z; \mathcal{B})}{2\epsilon z_l^k}, \quad (8)$$

where $z_l^k \in z$ is the random corresponding to W_l^k. In this way, the practical pruning score used in our MINI-LLM is defined as:

$$\hat{S}(W_l^k)_{ours} = \left| W_l^k \hat{g}_l^k X_l^k \right|. \quad (9)$$

Dependency-aware structured LLM pruning. To maintain structural integrity, it is crucial for structured pruning to identify groups of interdependent structures within LLMs. Following Ma et al. [2023], we prune heads for Multi-Head Attention (MHA) and channels for Feed-Forward Network (FFN), respectively. We arrange the interconnected weights into groups and determine the sensitivity of each group (a set of coupled structures) defined as $G = \{W_i\}_{i=1}^M$ by choosing the maximum sensitivity score of the structures in it, i.e., $\hat{S}(G) = \max_{i=1}^M \sum_k \hat{S}(W_i^k)$, where M is the number of interdependent structures in the group. Our structured pruning approach MINI-LLM is outlined in Algorithm 1.

3.3 Recovery with Low-rank Approximation

After pruning, we need a recovery stage to regain the performance. Due to the huge number of parameters, full fine-tuning becomes less feasible. LoRA [Hu *et al.*, 2022], as one of the most popular Parameter-Efficient Fine-Tuning (PEFT) methods [He *et al.*, 2023; Li and Liang, 2021; Jia *et al.*, 2022;

Chavan *et al.*, 2023; Lester *et al.*, 2021], has demonstrated strong capability for performance recovery, while significantly reducing GPU memory usage [Dettmers *et al.*, 2023].

To facilitate this, we fine-tune the pruned models by employing LoRA which only updates two injected low-rank decomposition matrices that are attached to a frozen pre-trained weight matrix. Given two low-rank matrices $A \in \mathbb{R}^{r \times k}$ and $B \in \mathbb{R}^{d \times r}$ ($r \ll \min(d, k)$), the forward computation can be written as:

$$f(x) = xW_0 + xBA, \quad (10)$$

where $x \in \mathbb{R}^{n \times d}$ denotes inputs. After adaption, the updated W can be re-parameterized as $W = W_0 + BA$.

4 Experiments

In this section, we evaluate the performance of our MINI-LLM on three kinds of LLMs, covering a wide range of tasks. We first introduce the experimental setup, then present the main results and provide ablation studies for further analysis.

4.1 Experimental Setup

Models, datasets, and evaluation metrics. To verify the effectiveness and versatility of our MINI-LLM, we test it over three open-source LLMs with different structures: LLaMA-7B [Touvron *et al.*, 2023], BLOOM-7B [Workshop, 2023], and OPT-6.7B [Zhang *et al.*, 2022]. All models undergo evaluation in a task-agnostic framework. We assess the zero-shot ability of pruned models on WikiText2 [Merity *et al.*, 2016] and PTB [Marcus *et al.*, 1993] for language generation with the perplexity (PPL)[2] analysis, and smaller is better. Besides, we follow LLaMA to implement zero-shot task classification and multiple-choice on four common sense reasoning datasets: BoolQ [Clark *et al.*, 2019], PIQA [Bisk *et al.*, 2020], HellaSwag [Zellers *et al.*, 2019], and Wino-Grande [Sakaguchi *et al.*, 2021]. In addition to zero-shot evaluation, we conduct experiments on few-shot tasks to evaluate pruned LLMs' ability to learn in context. We choose the Massive Multitask Language Understanding benchmark (**MMLU**) [[Hendrycks *et al.*, 2021]] and conduct a 5-shot evaluation to remain consistent with the evaluation approach described by [[Touvron *et al.*, 2023]]. In task classification and multiple-choice on common sense reasoning datasets, as well as on MMLU, classification accuracy is used as the performance metric.

Pruning and fine-tuning settings. Our MINI-LLM conducts in a one-shot pruning framework. That is scoring only once and then pruning the network to the target prune ratio [Cheng *et al.*, 2023]. In the model pruning process, we use 10 randomly selected samples from Bookcorpus [Zhu *et al.*, 2015] as the calibration data for evaluating the weight gradients and 128 samples for computing each layer's input (i.e., activation). Due to the varying sensitivity of each layer to pruning [Ma *et al.*, 2023], the first four layers and the last three layers are retained. During the recovery phase, we utilize the Alpaca-cleaned [Taori *et al.*, 2023] as the training dataset, which contains approximately 50k samples, to fine-tune the pruned models with a batch size of 64. Following [[Ma *et*

---

[2]https://huggingface.co/spaces/evaluate-metric/perplexity

Prune ratio	Method	GPU (GB)	WikiText2↓	PTB↓	BoolQ	PIQA	HellaSwag	WinoGrande	Average↑
0%	LLaMA-7B [Touvron *et al.*, 2023]*	0	-	-	76.50	79.8	76.10	70.10	75.63
	LLaMA-7B [Ma *et al.*, 2023]*	0	12.62	22.15	73.18	78.35	72.99	67.01	72.88
20% w/ tune	LLM-Pruner [Ma *et al.*, 2023]*	40.00	17.58	30.11	64.62	77.20	68.80	63.14	68.44
	magnitude-l1	21.60	24.32	43.19	58.47	75.35	65.40	60.93	65.04
	magnitude-l2	21.60	24.23	36.16	65.02	75.14	65.07	62.12	66.84
	SparseGPT [Frantar and Alistarh, 2023]	40.00	20.84	36.23	55.05	**75.84**	**66.88**	61.64	64.85
	Wanda [Sun *et al.*, 2024]	21.90	20.36	36.15	60.92	74.70	66.70	62.33	66.16
	MINI-LLM (ours)	22.40	**18.32**	**32.54**	**66.76**	75.46	65.61	**62.43**	**67.57**
30% w/ tune	LLM-Pruner [Ma *et al.*, 2023]	37.16	21.55	37.67	64.89	73.72	63.45	62.67	66.18
	magnitude-l1	20.10	31.17	54.28	61.80	73.23	58.02	56.91	62.49
	magnitude-l2	20.10	31.11	51.29	61.89	73.45	57.84	58.72	62.98
	SparseGPT [Frantar and Alistarh, 2023]	40.00	26.82	45.71	56.64	73.45	61.23	59.51	62.71
	Wanda [Sun *et al.*, 2024]	20.33	27.08	46.25	56.97	**74.27**	59.27	60.46	62.74
	MINI-LLM (ours)	20.90	**24.28**	**39.02**	64.55	73.74	58.74	58.93	**63.99**
40% w/ tune	LLM-Pruner [Ma *et al.*, 2023]	36.13	28.10	48.66	60.46	71.33	55.62	56.43	60.96
	magnitude-l1	18.80	43.96	66.63	47.03	70.89	49.79	52.96	55.17
	magnitude-l2	18.80	45.26	67.68	48.72	**71.65**	50.21	53.43	56.00
	SparseGPT [Frantar and Alistarh, 2023]	40.00	37.16	66.12	58.26	71.44	**53.91**	56.99	60.15
	Wanda [Sun *et al.*, 2024]	19.13	36.44	66.37	49.91	71.38	53.85	**58.72**	58.47
	MINI-LLM (ours)	19.83	**31.78**	**49.23**	**63.65**	71.59	53.31	55.56	**61.02**
50% w/ tune	LLM-Pruner [Ma *et al.*, 2023]*	35.00	38.12	66.35	60.28	69.31	47.06	53.43	57.52
	magnitude-l1	17.59	61.39	91.79	40.73	66.32	42.66	51.85	50.39
	magnitude-l2	17.59	58.12	89.67	38.50	67.08	43.47	52.80	50.46
	SparseGPT [Frantar and Alistarh, 2023]	40.00	49.52	82.28	42.84	68.01	45.38	**55.96**	53.05
	Wanda [Sun *et al.*, 2024]	18.82	45.98	78.82	40.49	**69.04**	45.10	54.93	52.39
	MINI-LLM (ours)	18.82	**44.69**	**69.83**	**61.35**	67.85	**45.39**	53.12	**56.93**

Table 1: Zero-shot performance of the pruned LLaMA-7B models. "Prune Ratio" refers to the proportion of parameters removed relative to the original number of parameters. "GPU (GB)" indicates the peak GPU memory usage for pruning. "Average" is calculated among four classification datasets. **Bold**/Underline mark the best/second best performance at the same compression rate with fine-tuning, respectively, excluding LLM-Pruner in the comparison. An asterisk (*) signifies the results are taken directly from the corresponding papers.

[1] We have saved the scores of each weight for SparseGPT, which results in high GPU memory usage. This can be optimized through implementation.

al., 2023]], the learning rate is set to 1e-4 and a total of 2 epochs. Each pruned model is recovered by an Adam optimizer [Kingma and Ba, 2015] paired with a cosine decay schedule for the learning rate. We set LoRA $r = 8$, $\alpha = 16$, and attach LoRA modules on all linear layers of the base model. In the inference stage, all the evaluations are implemented with a context length of 128.

Baselines. We compare MINI-LLM with four one-shot structured pruning methods for LLMs. Magnitude-l1/l2: pruning based on the absolute values or the l2-norm of weights, respectively. LLM-Pruner [Ma *et al.*, 2023]: pruning using criterion Eq. (3) with backpropagation gradients. Wanda [Sun *et al.*, 2024]: pruning based on the product of the magnitude of weights and their corresponding activations. Given that vanilla SparseGPT and Wanda are retraining-free unstructured methods, we adapt them for structured pruning with pruning and fine-tuning stages for a fair comparison while maintaining the same criterion. Except for LLM-pruner, which is a gradient-based method, the other methods are all gradient-free methods.

4.2 Main Results

Zero-shot performance on LLaMA-7B. We prune LLaMA-7B with four prune ratios: from 20% to 50% and fine-tune the pruned models by using LoRA to restore model accuracy. The comparisons with the baselines are reported in Table 1. From the results, we see that our MINI-LLM consistently surpasses all the gradient-free methods and closely matches or even outperforms the LLM-Pruner with backpropagation gradients across four prune ratios. For example, at a 20% prune ratio, MINI-LLM achieves an average classification accuracy of 67.57% across four inference datasets, better than other gradient-free methods, and obtains 92.71% of the accuracy achieved by the original model. Although LLM-Pruner achieves better accuracy with 93.91% of the accuracy attained by the dense model, the peak GPU memory required for pruning by LLM-Pruner is approximately twice that of MINI-LLM. Moreover, although both Wanda and MINI-LLM use weight magnitude and activation, MINI-LLM performs better, which indicates that estimated gradients are beneficial in guiding pruning. In addition, at a 40% prune ratio, MINI-LLM achieves an average accuracy of 61.02% on the four tasks, even better than LLM-Pruner's average accuracy of 60.96%.

However, similar to the observation in Ma et al. [2023], with a high prune ratio, such as 50%, an obvious performance decline is observed, as shown in Table 1. In this situation, our MINI-LLM and LLM-Pruner only retain 78.11% and 78.92% of the dense model's accuracy, respectively. Even for LLMs, structurally pruning under high prune ratios remains a major challenge.

Zero-shot performance on BLOOM-7B and OPT-6.7B. To validate MINI-LLM on other LLMs broadly, we prune both

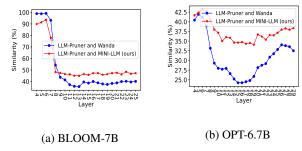|	
(a) BLOOM-7B	(b) OPT-6.7B

Figure 2: Similarity in pruned channels at the prune ratio of 30%. LLM-Pruner and MINI-LLM (ours) have more similar pruned channels compared to LLM-Pruner and Wanda.

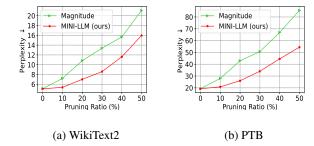|	
(a) WikiText2	(b) PTB

Figure 3: Zero-shot perplexity of the pruned LLaMA-13B models when fine-tuning is applied. MINI-LLM consistently maintains its substantial advantage over the magnitude-based method across a spectrum of pruning ratios.

BLOOM-7B and OPT-6.7B with two prune ratios: 10% and 30%, and fine-tune the pruned models to restore model accuracy. The results in Table 2 illustrate that our MINI-LLM steadily outperforms all gradient-free methods and exhibits performance comparable to, even surpasses at times, that of LLM-Pruner. For instance, at a 30% compression rate on BLOOM-7B, MINI-LLM achieves a perplexity of 54.07 on the WikiText2 dataset, obviously outperforming LLM-Pruner's perplexity of 58.11. Similarly, at a 30% compression rate on OPT-6.7B, MINI-LLM achieves a perplexity of 40.89 on the WikiText2 dataset and 57.44 on the PTB dataset, outperforming LLM-Pruner's perplexity of 42.94 and 65.09, respectively. In addition, at a 10% prune ratio on OPT-6.7B, MINI-LLM achieves an average classification accuracy of 67.81% across four datasets and obtains 98.60% of the accuracy achieved by the original model, which is even better than LLM-Pruner's 67.50% and 98.15%. This demonstration validates the effectiveness of MINI-LLM in efficiently compressing models of various structures to a specified size, while optimizing memory usage.

In addition, we observe that the pruning outcomes achieved by gradient-free methods such as Wanda and magnitude l1/l2 shown in Table 2 significantly fell short in comparison to gradient-based pruning methods such as LLM-Pruner and MINI-LLM at a prune ratio of 30% on the WikiText2 and PTB datasets for BLOOM and OPT. Using LLM-Pruner as a high-quality benchmark, we compare Wanda, representing gradient-free approaches, by assessing the similarity of their retained channels per layer against LLM-Pruner on the WikiText2 dataset. Similarly, we evaluate the similarity between LLM-Pruner and MINI-LLM. Specifically, the similarity is calculated by the formula: $||\text{Intersection}(A, B)||_0/||A||_0 \times 100\%$, where A and B denote the sets of the pruned channels obtained by LLM-Pruner and the examined method, respectively. The results are illustrated in Figure 2. We can see that LLM-Pruner and MINI-LLM have more similar pruned channels compared to LLM-Pruner and Wanda. As a result, compared to gradient-free methods, the perplexity of MINI-LLM in Table 2 is closer to the results of LLM-Pruner.

Zero-shot Performance on LLaMA-13B Due to the efficient approximation for the gradients of the pre-trained weights, MINI-LLM enables pruning on larger-scale LLMs,

such as LLaMA-13B [3]. We prune LLaMA-13B with five pruning ratios: from 10% to 50% and present the zero-shot performance of the pruned LLaMA-13B without fine-tuning in Table 3 and with fine-tuning in Figure 3, respectively. Except for the model pruned with a ratio of 50%, for which fine-tuning is conducted for two epochs, the compressed models obtained from all other pruning ratios are fine-tuned for just one epoch. The other fine-tuning settings, such as the learning rate and batch size, are the same as those for recovering LLaMA-7B. We follow Wanda [Sun *et al.*, 2024] to conduct inference with 2048 tokens.

As shown in Table 3, MINI-LLM outperforms the magnitude-based method (l2-norm) significantly when fine-tuning is not applied. For example, with a pruning ratio of 30%, MINI-LLM achieves a perplexity of 11.04 compared to the magnitude-based method's 316.65 on the WikiText2 dataset. Similarly, as depicted in Figure 3, MINI-LLM consistently maintains its substantial advantage over the magnitude-based method across a spectrum of pruning ratios when subjected to fine-tuning.

Few-shot performance on LLaMA-7B. In Table 4, we report the mean accuracies for both dense LLMs and sparse LLMs with 20% to 50% sparsity. In the few-shot setting, MINI-LLM performs competitively with other methods, including backpropagation gradient-based LLM-Pruner. Specifically, at a 20% prune ratio, MINI-LLM achieves an average accuracy of 26.60%, which surpasses SparseGPT's 25.80% and LLM-Pruner's 25.30%. Notably, estimated gradient-based MINI-LLM consistently surpasses backpropagation gradient-based LLM-Pruner. This performance is not observed in zero-shot setting.

Model size, complexity, and inference time. Table 5 shows the number of parameters, MACs, GPU memory requirements, and total inference time for running the original model and the pruned LLaMA-7B models at different prune ratios. The results indicate that when the model is pruned by 50%, the total inference time is reduced to 58% and the GPU memory usage concurrently drops to 50% of its original values, respectively. The evaluation is conducted in the inference

---

[3]https://huggingface.co/huggyllama/llama-13b/tree/main

Prune ratio	Method	GPU (GB)	WikiText2↓	PTB↓	BoolQ	PIQA	HellaSwag	WinoGrande	Average↑
0%	BLOOM-7B [Workshop, 2023]	0	26.58	50.55	62.94	73.61	59.69	64.4	65.16
10% w/ tune	LLM-Pruner [Ma et al., 2023]	40.00	35.32	75.21	62.14	72.14	55.39	57.85	61.88
	magnitude-l1	22.04	40.89	92.87	59.28	71.82	52.44	56.21	59.94
	magnitude-l2	22.04	40.73	95.45	59.33	72.04	52.58	56.04	60.00
	SparseGPT [Frantar and Alistarh, 2023]	40.00	40.42	92.15	59.17	70.73	52.38	56.2	59.62
	Wanda [Sun et al., 2024]	22.81	40.81	93.60	59.94	72.25	52.60	57.14	60.48
	MINI-LLM (ours)	25.03	38.12	86.23	59.97	72.05	53.54	56.43	60.50
30% w/ tune	LLM-Pruner [Ma et al., 2023]	38.51	58.11	147.52	62.11	67.79	44.04	53.28	56.81
	magnitude-l1	19.49	87.25	166.21	61.04	65.40	41.46	51.70	54.90
	magnitude-l2	19.49	79.75	167.83	59.45	66.87	42.36	50.91	54.89
	SparseGPT [Frantar and Alistarh, 2023]	40.00	75.51	173.51	52.02	67.14	42.86	53.28	53.83
	Wanda [Sun et al., 2024]	20.34	84.89	170.16	53.61	67.03	41.34	50.99	53.24
	MINI-LLM (ours)	22.11	54.07	121.61	62.17	68.82	44.95	51.93	56.97
0%	OPT-6.7B [Zhang et al., 2022]	0	26.45	32.03	66.06	76.55	67.21	65.27	68.77
10% w/ tune	LLM-Pruner [Ma et al., 2023]	38.00	27.89	39.33	63.06	76.77	66.33	63.85	67.50
	magnitude-l1	22.81	39.17	55.68	58.17	75.30	60.35	59.59	63.35
	magnitude-l2	22.81	39.40	54.49	59.08	74.86	60.45	60.22	63.65
	SparseGPT [Frantar and Alistarh, 2023]	40.00	36.58	50.99	61.93	74.81	61.25	60.46	64.61
	Wanda [Sun et al., 2024]	23.04	37.09	53.54	66.09	75.46	62.24	62.59	66.60
	MINI-LLM (ours)	23.65	30.15	38.64	65.90	76.12	65.66	63.54	67.81
30% w/ tune	LLM-Pruner [Ma et al., 2023]	34.66	42.94	65.09	61.93	73.83	56.98	59.98	63.60
	magnitude-l1	19.91	81.96	104.01	48.50	69.59	44.99	53.75	54.21
	magnitude-l2	19.91	76.10	98.86	54.22	69.37	44.83	54.14	56.11
	SparseGPT [Frantar and Alistarh, 2023]	40.00	77.00	103.61	54.31	69.21	44.56	55.56	55.91
	Wanda [Sun et al., 2024]	20.07	82.93	107.32	60.34	69.53	44.58	54.46	57.23
	MINI-LLM (ours)	20.65	40.89	57.44	62.17	72.58	54.07	56.20	61.26

Table 2: Zero-shot performance of the pruned BLOOM-7B and OPT-6.7B. Columns is consistent with the definitions in Table 1. Unless otherwise specified, "Prune Ratio" and **Bold**/<u>Underline</u> have the same meaning as Table 1.

Method	Dataset	Prune Ratio					
		0%	10%	20%	30%	40%	50%
Magnitude (l2-norm)	WikiText2	**5.09**	13.78	21.42	316.65	3918.40	12550.30
MINI-LLM		**5.09**	**5.87**	**7.51**	**11.04**	**22.32**	**115.13**
Magnitude (l2-norm)	PTB	**19.24**	50.89	91.87	694.34	4236.80	12847.92
MINI-LLM		**19.24**	**23.86**	**33.38**	**49.91**	**91.43**	**176.93**

Table 3: Zero-shot perplexity of the pruned LLaMA-13B when fine-tuning is not applied.

Ratio	Method	MMLU (5-shot)				
		STEM	Humans	Social	Other	Avg.
0%	-	32.60	34.10	40.40	40.90	36.70
20%	SparseGPT [2023]	<u>25.30</u>	**25.90**	<u>25.50</u>	26.30	<u>25.80</u>
	Wanda [2024]	23.30	<u>25.80</u>	23.20	24.60	24.40
	LLM-Pruner [2023]	24.40	25.30	23.80	<u>27.30</u>	25.30
	MINI-LLM (ours)	**25.50**	**25.90**	**26.20**	**29.00**	**26.60**
30%	SparseGPT [2023]	**25.80**	<u>25.70</u>	24.60	23.50	25.00
	Wanda [2024]	**25.80**	**27.10**	<u>25.00</u>	24.80	**25.80**
	LLM-Pruner [2023]	23.90	24.90	23.50	<u>26.00</u>	24.60
	MINI-LLM (ours)	<u>24.10</u>	24.80	**25.70**	**26.20**	<u>25.20</u>
40%	SparseGPT [2023]	<u>26.10</u>	**25.50**	23.30	23.90	24.80
	Wanda [2024]	25.80	<u>24.50</u>	<u>24.80</u>	23.60	<u>25.80</u>
	LLM-Pruner [2023]	22.70	24.40	21.40	<u>24.00</u>	23.30
	MINI-LLM (ours)	**26.20**	24.10	**27.50**	**27.50**	**26.10**
50%	SparseGPT [2023]	**26.40**	24.70	**25.40**	24.20	<u>25.10</u>
	Wanda [2024]	26.00	**25.10**	24.30	<u>25.00</u>	<u>25.10</u>
	LLM-Pruner [2023]	21.30	24.20	21.70	23.70	22.90
	MINI-LLM (ours)	<u>26.30</u>	<u>24.80</u>	<u>25.00</u>	25.30	25.30

Table 4: Few-shot performance of the pruned LLaMA-7B models. "Ratio", **Bold**/<u>Underline</u>, "Avg." have the same meaning as Table 1

mode and the sequence length is set to 64. The inference time is tested under the test dataset of WikiText2 on a single NVIDIA GeForce RTX 3090Ti (24GB).

4.3 Ablation Study

Efficacy of estimated gradients on LLaMA-7B. To enhance GPU memory efficiency over traditional backpropagation gradients, we utilize the classical ZO gradient estimation based on SPSA to approximately compute weight gradients with only forward passes for LLM pruning. Although SPSA-based ZO optimization is theoretically founded ([Spall, 1992; Spall, 1997; Gasnikov et al., 2022]), we especially reveal the effectiveness of the estimated gradients for guiding pruning LLMs in Figure 4. As we can see, the results demonstrate that our score function FMS, $\left| W \hat{\nabla} \mathcal{L}(W) X \right|$ (represented by the red line in Figure 4a), consistently yields better performance compared to Wanda's pruning criterion, $|WX|$ (indicated by the green line), across four prune ratios ranging from 20% to 50% for LLaMA-7B on the WikiText2 dataset. On the PTB dataset, this improved performance is more evident, as shown in Fig 4b. Comparing these two criteria, our FMS includes an additional estimated gradient $\hat{\nabla} \mathcal{L}(W)$ compared to Wanda's. This indicates that the performance improvement over Wanda's comes from the estimated gradient information. This observation underscores the superior performance and

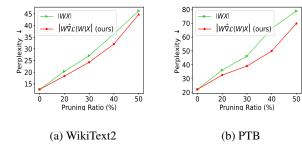(a) WikiText2   (b) PTB

Figure 4: The outcomes of gradient-based vs. gradient-free criteria for pruning LLaMA-7B. The results demonstrate that our score function FMS consistently yields better performance compared to Wanda's pruning criterion.

effectiveness of gradient-based pruning methods in our experiments. However, as we previously mentioned, gradients based on backpropagation lead to substantial memory consumption and are less feasible. Therefore, gradient estimation based on the forward passes becomes valuable, allowing the criterion to incorporate guidance information from gradients.

Ratio	#Params	#MACs	GPU Memory	Inference Time
0%	6.74B	424.02G	12884.5MB	88.81s
20%	5.42B	340.48G	10375.5MB	71.77s
50%	3.39B	279.37G	6519.0MB	51.18s

Table 5: Model size, complexity, and inference time of the original model and the pruned LLaMA-7B models. "Inference Time" means the total inference time on WikiText2 test dataset. The evaluation is conducted in inference mode with a sequence length of 64, and the inference time is tested on a single NVIDIA GeForce RTX 3090 Ti (24GB).

Efficacy of Estimated Gradients on BLOOM-7B and OPT-6.7B In the main body, we explored the effectiveness of the estimated gradients for guiding the pruning of LLaMA-7B. Here, we delve further into the effectiveness of the estimated gradients in guiding the pruning process for BLOOM-7B [4] and OPT-6.7B [5]. The experimental results presented in Table 6 indicate that our score function FMS, $|W\hat{\nabla}\mathcal{L}(W)X|$, consistently yields better performance compared to Wanda's pruning criterion, $|WX|$, for pruning BLOOM-7B and OPT-6.7B on the WikiText2 and PTB datasets. For example, at a 30% pruning ratio, $|W\hat{\nabla}\mathcal{L}(W)X|$ achieves a perplexity of 54.07 on WikiText2 for pruning BLOOM-7B. This result shows a 30.82 improvement over $|WX|$. Similarly, at a 30% pruning ratio, $|W\hat{\nabla}\mathcal{L}(W)X|$ achieves a perplexity of 40.89 on WikiText2 for pruning OPT-6.7B, which surpasses the perplexity of 82.93 obtained by using $|WX|$.

Efficacy of activation on LLaMA-7B. [Dettmers *et al.*, 2022; Kovaleva *et al.*, 2021] identified a distinct property of LLMs that a few hidden state features possess notably high magnitudes. Eliminating these features results in a considerable decline in performance. As argued in Section 3.1, the

[4]https://huggingface.co/bigscience/bloom-7b1/tree/main
[5]https://huggingface.co/facebook/opt-6.7b/tree/main

Prune Ratio	Model/Criterion	WikiText2↓	Δ ↑	PTB↓	Δ ↑
0%	BLOOM-7B	26.58	-	50.55	-
10%	$|WX|$	40.81	2.69	93.60	7.37
w/ tune	$|W\hat{\nabla}\mathcal{L}(W)X|$	**38.12**		**86.23**	
30%	$|WX|$	84.89	30.82	170.16	48.55
w/ tune	$|W\hat{\nabla}\mathcal{L}(W)X|$	**54.07**		**121.61**	
0%	OPT-6.7B	26.45	-	32.03	-
10%	$|WX|$	37.09	6.94	53.54	14.90
w/ tune	$|W\hat{\nabla}\mathcal{L}(W)X|$	**30.15**		**38.64**	
30%	$|WX|$	82.93	42.04	107.32	49.88
w/ tune	$|W\hat{\nabla}\mathcal{L}(W)X|$	**40.89**		**57.44**	

Table 6: The outcomes of gradient-based vs. gradient-free criteria for pruning BLOOM-7B and OPT-6.7B. Δ represents the difference in perplexity between the pruning criterion without gradients and the one with gradients. A larger Δ value indicates a greater improvement in performance.

vanilla pruning criterion $|W\nabla\mathcal{L}(W)|$ (Eq. (3)) does not highlight this characteristic of LLMs. To validate that activation in FMS, i.e., $|W\nabla\mathcal{L}(W)X|$ (Eq. (6)), can bring improved performance, we compare the performance of the pruned models obtained by using the criteria Eq. (3) and Eq. (6) over four prune ratios on the LLM. The difference between these two criteria is that Eq. (6) includes an additional activation term X compared to Eq. (3). From the results shown in Table 7, we can see that the activations enable an effective increase in performance on both the WikiText2 and PTB datasets. For example, at a 20% prune ratio, the model pruned with $|W\nabla\mathcal{L}(W)X|$ achieves a perplexity of 17.45 on the WikiText2 dataset. This result surpasses the 17.79 perplexity achieved by the model compressed by using $|W\nabla\mathcal{L}(W)|$. In contrast, on the PTB dataset, the performance enhancement provided by activation is generally more noticeable than on WikiText2. For instance, with 50% parameters pruned, the model pruned with $|W\nabla\mathcal{L}(W)X|$ achieves a perplexity of 62.84 on the PTB dataset. Compared to using $|W\nabla\mathcal{L}(W)|$, the perplexity result decreased by 3.53. It is worth noting that activations can be estimated using a small set of calibration data and executed in a single forward pass. Therefore, computing $|W\nabla\mathcal{L}(W)X|$, as opposed to $|W\nabla\mathcal{L}(W)|$, almost does not require additional GPU memory overhead.

In Table 7, gradients in both pruning criteria are calculated by backpropagation. In contrast, the results in Table 8 demonstrate the effectiveness of activation in estimated gradients-based pruning criteria. Similar to Table 7, the difference between the two criteria in Table 8 also lies in whether they include activation information or not. The results in Table 8 indicate that using the pruning criterion with activation consistently yielded better results. For example, on the WikiText2 dataset, the model pruned with 50% of its parameters using $|W\hat{\nabla}\mathcal{L}(W)X|$ achieves a perplexity of 44.69, reflecting a 8.54 improvement over the 53.23 perplexity by the model using $|W\hat{\nabla}\mathcal{L}(W)|$. Switching to the PTB dataset on the same compression rate, also yielded positive results, with the model's perplexity dropping from 75.50 to 69.83, confirming the efficacy of the activation-inclusive pruning crite-

Ratio	Criterion	WikiText2↓	Δ↑	PTB↓	Δ↑	Avg.↑	Δ↑
20%	$\|W\nabla\mathcal{L}(W)\|$	17.79	0.34	**30.57**	-0.12	68.44	0.91
	$\|W\nabla\mathcal{L}(W)X\|$	**17.45**		30.69		**69.35**	
30%	$\|W\nabla\mathcal{L}(W)\|$	21.55	0.38	37.67	0.80	66.18	0.39
	$\|W\nabla\mathcal{L}(W)X\|$	**21.17**		**36.87**		**66.57**	
40%	$\|W\nabla\mathcal{L}(W)\|$	28.10	0.08	48.66	2.05	60.96	1.11
	$\|W\nabla\mathcal{L}(W)X\|$	**28.02**		**46.61**		**62.07**	
50%	$\|W\nabla\mathcal{L}(W)\|$	39.48	0.38	66.37	3.53	57.52	0.94
	$\|W\nabla\mathcal{L}(W)X\|$	**39.10**		**62.84**		**58.46**	

Table 7: Zero-shot performance of the pruned LLaMA-7B models achieved by using backpropagation gradient-based pruning criterion with/without activations. "Ratio" refers to the prune ratio. Δ represents the difference in performance between the pruning criterion without activation and the one with activation. A larger Δ value indicates a greater improvement in performance. "Avg." has the same meaning as "Average" in Table 1.

Ratio	Criterion	WikiText2↓	Δ↑	PTB↓	Δ↑	Avg.↑	Δ↑
20%	$\|W\hat{\nabla}\mathcal{L}(W)\|$	20.40	2.08	34.17	1.63	67.24	0.33
	$\|W\hat{\nabla}\mathcal{L}(W)X\|$	**18.32**		**32.54**		**67.57**	
30%	$\|W\hat{\nabla}\mathcal{L}(W)\|$	25.09	0.81	40.89	1.87	61.97	2.02
	$\|W\hat{\nabla}\mathcal{L}(W)X\|$	**24.28**		**39.02**		**63.99**	
40%	$\|W\hat{\nabla}\mathcal{L}(W)\|$	35.25	3.47	52.30	3.07	58.95	2.07
	$\|W\hat{\nabla}\mathcal{L}(W)X\|$	**31.78**		**49.23**		**61.02**	
50%	$\|W\hat{\nabla}\mathcal{L}(W)\|$	53.23	8.54	75.50	5.67	54.73	2.20
	$\|W\hat{\nabla}\mathcal{L}(W)X\|$	**44.69**		**69.83**		**56.93**	

Table 8: Zero-shot performance of the pruned LLaMA-7B models achieved by using ZO gradient-based pruning criterion with/without activations. The columns have the same meaning as Table 7.

rion across diverse data.

Consequently, the results in Table 7 and Table 8 demonstrate our pruning criterion FWS overall outperforms its counterpart without activation, whether the gradients are backpropagated or approximated.

Efficacy of activations on BLOOM-7B and OPT-6.7B Table 9, Table 10 and Table 11 show the zero-shot results of the pruned models with/without fine-tuning to discover the effectiveness of activations for guiding pruning LLMs. From the results in Table 9, it is evident that incorporating activations $|W\nabla\mathcal{L}(W)X|$ results in an overall enhancement of performance compared to the standard pruning criterion $|W\nabla\mathcal{L}(W)|$. For example, at a 10% pruning ratio, the model pruned with $|W\nabla\mathcal{L}(W)X|$ achieves a perplexity of 35.53 on the WikiText2 dataset for pruning BLOOM-7B. This result surpasses the 38.12 perplexity achieved by the model compressed by using $|W\nabla\mathcal{L}(W)|$. For OPT-6.7B, the performance enhancement provided by activation is also noticeable. For instance, with 30% parameters pruned, the model pruned with $|W\nabla\mathcal{L}(W)X|$ achieves a perplexity of 64.58 on PTB. Compared to using $|W\nabla\mathcal{L}(W)|$, the perplexity result decreased by 0.51.

In Table 9, gradients are computed by backpropagation. In contrast, the results presented in Table 10 and 11 highlight the effectiveness of incorporating activations in estimated gradient-based pruning criteria when fine-tuning is applied or not. For instance, when fine-tuning is not applied, the model pruned with 30% of BLOOM-7B's parameters using

Prune Ratio	Model/Criterion	WikiText2↓	Δ↑	PTB↓	Δ↑
0%	BLOOM-7B	26.58	-	50.55	-
10% w/ tune	$\|W\nabla\mathcal{L}(W)\|$	38.12	2.59	86.23	8.63
	$\|W\nabla\mathcal{L}(W)X\|$	**35.53**		**77.60**	
30% w/ tune	$\|W\nabla\mathcal{L}(W)\|$	**58.11**	-0.24	147.52	9.72
	$\|W\nabla\mathcal{L}(W)X\|$	58.35		**137.80**	
0%	OPT-6.7B	26.45	-	32.03	-
10% w/ tune	$\|W\nabla\mathcal{L}(W)\|$	**27.89**	-0.38	39.33	0.39
	$\|W\nabla\mathcal{L}(W)X\|$	28.27		**38.94**	
30% w/ tune	$\|W\nabla\mathcal{L}(W)\|$	42.94	0.75	65.09	0.51
	$\|W\nabla\mathcal{L}(W)X\|$	**42.19**		**64.58**	

Table 9: Zero-shot perplexity of the pruned BLOOM-7B and OPT-6.7B models achieved by using backpropagation gradient-based pruning criterion with/without activations. The columns have the same meaning as Table 6.

Prune Ratio	Model/Criterion	WikiText2↓	Δ↑	PTB↓	Δ↑
0%	BLOOM-7B	26.58	-	50.55	-
10% w/o tune	$\|W\hat{\nabla}\mathcal{L}(W)\|$	78.21	4.74	231.67	27.22
	$\|W\hat{\nabla}\mathcal{L}(W)X\|$	**73.47**		**204.45**	
30% w/o tune	$\|W\hat{\nabla}\mathcal{L}(W)\|$	106.07	14.64	239.96.52	31.48
	$\|W\hat{\nabla}\mathcal{L}(W)X\|$	**91.43**		**208.48**	

Table 10: Zero-shot perplexity of the pruned BLOOM-7B achieved by using ZO gradient-based pruning criterion with/without activations. The columns have the same meaning as Table 6.
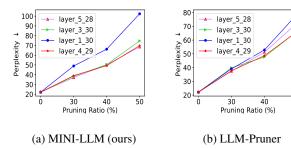
$|W\hat{\nabla}\mathcal{L}(W)X|$ achieves a perplexity of 91.43 on WikiText2, reflecting a 14.64 improvement over the 106.07 perplexity by the model using $|W\hat{\nabla}\mathcal{L}(W)|$. In contrast, after undergoing fine-tuning, although the increase in performance becomes less pronounced as shown in Table 11, it is still evident that activations play a significant role in enhancing performance.

Layer Sensitivity for Pruning Based on the findings in [[Ma et al., 2023]] that the first and last layers significantly affect the model's performance, we investigate the impact of involving different ranges of layers in the pruning process on LLaMA-7B's performance. It includes analyzing the performance of models with pruning applied from the 1st to the 30th layer (represented by layer-1-30 in Figure 5), from the 3rd to the 30th layer (layer-3-30), from the 4th to the 29th layer (layer-4-29), and from the 5th to the 28th layer (layer-5-28) [6]. From the results in Figure 5, it is evident that layer-1-30 has the worst performance, while layer-3-30 and layer-4-29 have comparably better performance for both pruning methods. In contrast, the models derived from layer-5-28 pruning exhibit varying responses to different pruning methods. For instance, for MINI-LLM, their performance is similar to that of the layer-4-29 models, whereas, for LLM-Pruner [[Ma et al., 2023]], their performance is somewhat inferior compared to both layer-4-29 and layer-3-30 models. Since the layer-4-29 pruning demonstrates consistent performance across various pruning methods, we conduct pruning for layer-4-29 in all LLaMA-7B experiments.

Considering the inferior performance of layer-1-31, in Fig-

---

[6]LLaMA-7B has 32 layers, from 0th to the 31st layer.

Prune Ratio	Model/Criterion	WikiText2↓	Δ ↑	PTB↓	Δ ↑
0%	OPT-6.7B	26.45	-	32.03	-
10%	$\lvert W\hat{\nabla}\mathcal{L}(W)\rvert$	30.51		39.17	
w/ tune	$\lvert W\hat{\nabla}\mathcal{L}(W)X\rvert$	**30.15**	0.36	**38.64**	0.53
30%	$\lvert W\hat{\nabla}\mathcal{L}(W)\rvert$	43.87		58.12	
w/ tune	$\lvert W\hat{\nabla}\mathcal{L}(W)X\rvert$	**40.89**	2.98	**57.44**	0.68

Table 11: Zero-shot perplexity of the pruned OPT-6.7B achieved by using ZO gradient-based pruning criterion with/without activations. The columns have the same meaning as Table 6.

(a) MINI-LLM (ours)　　　　(b) LLM-Pruner

Figure 5: The zero-shot perplexity of the pruned models achieved by enabling different ranges of layers involved in pruning LLaMA-7B on the PTB dataset. Layer-1-30 has the worst performance, while layer-3-30 and layer-4-29 have comparably better performance for both pruning methods. In contrast, the models derived from layer-5-28 pruning exhibit varying responses to different pruning methods.

ure 6, we only display the proportions of layers involved in pruning for layer-3-30, layer-4-29, and layer-5-28. In addition, we present the average perplexity of LLM-Pruner and MINI-LLM for the three ranges of layers, marked with the orange or the blue five-pointed stars in Figure 6, which is averaged over three pruning ratios: 30%, 40%, and 50 % on Wiki-Text2 and PTB. The average perplexity results in Figure 6 illustrate that our MINI-LLM exhibits performance close to, even surpasses at times, that of LLM-Pruner. For instance, for layer-5-28, MINI-LLM achieves an average perplexity of 51.87 on PTB, outperforming LLM-Pruner's 53.73. These results demonstrate that MINI-LLM is a memory-efficient and effective method for gradient-based pruning.

4.4 Generations From Pruned Model

Table 12 shows the generation examples of the original and the pruned LLaMA-7B models achieved by MINI-LLM. The five experimental instructions encompass math, common sense, translation, and writing tasks. From the responses presented in Table 12, it is evident that when pruning 20% of the parameters, the pruned model maintains high performance in these tasks.

5 Conclusion

In this paper, we presented MINI-LLM, an one-shot structured pruning approach designed to address the high GPU memory demands of computing backpropagation-based gradients of pre-trained LLMs. First, we proposed a novel criterion called the Feature Map Sensitivity (FMS) score which integrates magnitude, activation, and gradient information to
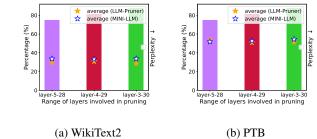

(a) WikiText2　　　　　　(b) PTB

Figure 6: The percentage of different ranges of layers involved in pruning LLaMA-7B and the average perplexity. MINI-LLM exhibits performance close to, even surpasses LLM-Pruner.

guide the pruning process effectively. By employing estimated gradients based on forward passes, MINI-LLM not only reduces the GPU memory requirement for gradient-guided pruning but also achieves superior performance compared to existing gradient-free methods. Our extensive experiments on three LLMs: LLaMA, BLOOM, and OPT, across various downstream tasks demonstrate MINI-LLM's effectiveness and efficiency in GPU memory usage. In the future, our objective is to further enhance the pruning results of MINI-LLM at higher compression rates.

References

[Bisk et al., 2020] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *AAAI*, 2020.

[Brown et al., 2020] Tom Brown, Benjamin Mann, Nick Ryder, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[Chavan et al., 2023] Arnav Chavan, Zhuang Liu, Deepak Gupta, Eric Xing, and Zhiqiang Shen. One-for-All: Generalized lora for parameter-efficient fine-tuning. *arXiv preprint arXiv:2306.07967*, 2023.

[Cheng et al., 2023] Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. A survey on deep neural network pruning-taxonomy, comparison, analysis, and recommendations. *arXiv preprint arXiv:2308.06767*, 2023.

[Clark et al., 2019] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *NAACL*, 2019.

[Dettmers et al., 2022] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. LLM.int8(): 8-bit matrix multiplication for transformers at scale. In *NeurIPS*, 2022.

[Dettmers et al., 2023] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. *arXiv preprint arXiv:2305.14314*, 2023.

[Frantar and Alistarh, 2023] Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately

Model	Generated Sentences
LLaMA-7B (ratio=0%)	<u>Instruction:</u> What are Fibonacci numbers? <u>Response:</u> Fibonacci numbers is a sequence of integer numbers, generated by a simple formula which has the form of sum of two numbers. This sequence has a few very interesting properties which can be summarized as follows: Every Fibonacci number is the sum of two preceding Fibonacci numbers. The formula is Fn= Fn-1 + Fn-2. Start with 1,2,3,5,8,13,21,34,55,89,...
	<u>Instruction:</u> What is the difference between a cat and a tiger? <u>Response:</u> A tiger is a member of the cat family, but is a big, powerful animal with striped fur. It is bigger than a normal housecat. The tiger is native to India and southwestern Asia. In contrast, a cat is smaller than the tiger. There are three main types of cats. Most cats are of the domestic.
	<u>Instruction:</u> AI can create a logo in seconds <u>Response:</u> When you think about a logo, the first thought may be to sit down and draw something, or even create a vector file. But the truth is that it is possible to do this easily and quickly by using AI. So, the question is, why not use it? For example, I was recently working with the logo of an Italian company and I only needed to sit down at the computer for 5 minutes to create the basic elements that I would then change to meet the needs of the brand. In just 2 minutes you can create a 3D logo.
	<u>Instruction:</u> Compose a poem about summer. <u>Response:</u> Sandy toes, Cooling ocean, Water lilies, the warm sun, Making the clouds, dissolve, and the birds sing. Their songs lull me to sleep, With soft breeze. Summer is, the best, because there is no one to bother, in the summer days.
	<u>Instruction:</u> Translate the English sentence "large language models" into Spanish. <u>Response:</u> The answer for the English sentence is "grandes modelos lenguas" in Spanish.
LLaMA-7B (ratio=20%)	<u>Instruction:</u> What are Fibonacci numbers? <u>Response:</u> In mathematics and physics, the Fibonacci numbers are the sequence of integers starting from the 2nd number as the sum of the previous two numbers in the sequence: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55,...
	<u>Instruction:</u> What is the difference between a cat and a tiger? <u>Response:</u> The cat and the tiger are both Felines, which means both species belong to the same Biological Taxonomic Classification family. However, the difference between a cat and a tiger lies primarily in the physical differences between the two species. A cat is generally smaller in size, with an average weight of 5 - 10 pounds (2.7 - 4.4 kilograms). Cats are more domesticated, often being kept as pets. They are known for being more social, and their distinctive meowing, hisses, and growls are used to communicate with other felines. On the other hand, tigers are bigger in size, with an average weight of around 125 pounds (69.5 kilograms). They are also more wild, with little domestication.
	<u>Instruction:</u> AI can create a logo in seconds <u>Response:</u> AI can create a logo in seconds, a visual designer can bring your brand idea to life. When it comes to creating a brand identity, our first step is to conduct a thorough market analysis and develop a brand position. From there we can create a creative brand identity that is in line with the core values of your business. Our creative design team is trained in building brands that are aesthetically pleasing, easy to recognize and communicate a clear message to your target audience. The design can incorporate many elements including illustrations, photography, infographics, videos, typography and layout.
	<u>Instruction:</u> Compose a poem about summer. <u>Response:</u> A summer sun is all aglow, Reflecting in the calm, clear waters below, Tossed by gentle breeze, Warm and golden, Far beyond sight. A perfect sight for sight, In an endless horizon, Reflecting in the calm, Clear waters below. A summer sun, That never sets, Far beyond sight.
	<u>Instruction:</u> Translate the English sentence "large language models" into Spanish. <u>Response:</u> Here's a suggested sentence in Spanish: "Modelo grande de lenguas."

Table 12: Generated Examples from the original and pruned LLaMA-7B.

pruned in one-shot. *arXiv preprint arXiv:2301.00774*, 2023.

[Frantar *et al.*, 2022] Elias Frantar, Sidak Pal Singh, and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. In *NeurIPS*, 2022.

[Fu *et al.*, 2022] Yonggan Fu, Haichuan Yang, Jiayi Yuan, Meng Li, Cheng Wan, Raghuraman Krishnamoorthi, Vikas Chandra, and Yingyan Lin. Depthshrinker: a new compression paradigm towards boosting real-hardware efficiency of compact neural networks. In *ICML*, 2022.

[Gasnikov *et al.*, 2022] Alexander Gasnikov, Darina Dvinskikh, Pavel Dvurechensky, Eduard Gorbunov, Aleksander Beznosikov, and Alexander Lobanovu. Randomized gradient-free methods in convex optimization. *arXiv preprint arXiv:2211.13566*, 2022.

[He *et al.*, 2023] Haoyu He, Jianfei Cai, Jing Zhang, Dacheng Tao, and Bohan Zhuang. Sensitivity-aware visual parameter-efficient tuning. In *ICCV*, 2023.

[Hendrycks *et al.*, 2021] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations (ICLR)*, 2021.

[Hu *et al.*, 2022] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR poster*, 2022.

[Jia *et al.*, 2022] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *ECCV*, 2022.

[Kiefer and Wolfowitz., 1952] J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *Ann. Math. Statist*, 23:462–466, 1952.

[Kingma and Ba, 2015] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[Kovaleva *et al.*, 2021] Olga Kovaleva, Saurabh Kulshreshtha, Anna Rogers, and Anna Rumshisky. BERT

Busters: Outlier dimensions that disrupt transformers. In *ACL*, 2021.

[Kurtic *et al.*, 2022] Eldar Kurtic, Daniel Campos, Tuan Nguyen, Elias Frantar, Mark Kurtz, Benjamin Fineran, Michael Goin, and Dan Alistarh. The optimal BERT surgeon: Scalable and accurate second-order pruning for large language models. In *EMNLP*, 2022.

[Kurtic *et al.*, 2023] Eldar Kurtic, Elias Frantar, and Dan Alistarh. ZipLM: Inference-aware structured pruning of language models. In *NeurIPS*, 2023.

[Kwon *et al.*, 2022] Woosuk Kwon, Sehoon Kim, Michael W. Mahoney, Joseph Hassoun, Kurt Keutzer, and Amir Gholami. A fast post-training pruning framework for transformers. In *NeurIPS*, 2022.

[LeCun *et al.*, 1989] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. In *NIPS*, pages 598–605, 1989.

[Lee *et al.*, 2019] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. SNIP: Single-shot network pruning based on connection sensitivity. In *ICLR*, 2019.

[Lester *et al.*, 2021] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *EMNLP*, 2021.

[Li and Liang, 2021] Xiang Lisa Li and Percy Liang. Prefix-Tuning: Optimizing continuous prompts for generation. In *IJCNLP*, 2021.

[Li *et al.*, 2020] Yawei Li, Shuhang Gu, Christoph Mayer, Luc Van Gool, and Radu Timofte. Group sparsity: The hinge between filter pruning and decomposition for network compression. In *CVPR*, 2020.

[Li *et al.*, 2022a] Shiru Li, Yong Xia, and Zi Xu. Simultaneous perturbation stochastic approximation: towards one-measurement per iteration. *arXiv preprint arXiv:2203.03075*, 2022.

[Li *et al.*, 2022b] Yuchao Li, Fuli Luo, Chuanqi Tan, Mengdi Wang, Songfang Huang, Shen Li, and Junjie Bai. Parameter-efficient sparsity for large language models fine-tuning. In *IJCAI*, 2022.

[Liu *et al.*, 2021] Liyang Liu, Shilong Zhang, Zhanghui Kuang, Aojun Zhou, Jing-Hao Xue, Xinjiang Wang, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. Group fisher pruning for practical network compression. In *ICML*, 2021.

[Ma *et al.*, 2023] Xinyin Ma, Gongfan Fang, and Xinchao Wang. LLM-Pruner: On the structural pruning of large language models. In *NeurIPS*, 2023.

[Malladi *et al.*, 2023] Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D. Lee, Danqi Chen, and Sanjeev Arora. Fine-tuning language models with just forward passes. In *NeurIPS*, 2023.

[Marcus *et al.*, 1993] Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The penn treebank. *Computational Linguistics*, 19:313–330, 1993.

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

[Molchanov *et al.*, 2017] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource-efficient inference. In *ICLR*, 2017.

[Molchanov *et al.*, 2019] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *CVPR*, 2019.

[Nova *et al.*, 2023] Azade Nova, Hanjun Dai, and Dale Schuurmans. Gradient-free structured pruning with unlabeled data. In *ICML*, 2023.

[OpenAI, 2023] OpenAI. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[Sakaguchi *et al.*, 2021] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande:an adversarial winograd schema challenge at scale. *Communications of the ACM*, 64:99–106, 2021.

[Sanh *et al.*, 2020] Victor Sanh, Thomas Wolf, and Alexander M. Rush. Movement pruning: Adaptive sparsity by fine-tuning. In *NeurIPS*, 2020.

[Spall, 1992] James C. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37:332–341, 1992.

[Spall, 1997] James C. Spall. A one-measurement form of simultaneous perturbation stochastic approximation. *Automatics*, 33:109–112, 1997.

[Sun *et al.*, 2024] Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models. In *Proceedings of International Conference on Learning Representations (ICLR) poster*, 2024.

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

[Touvron *et al.*, 2023] Hugo Touvron, Thibaut Lavril, Gautier Izacard, et al. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[Wang *et al.*, 2020a] Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. In *ICLR*, 2020.

[Wang *et al.*, 2020b] Ziheng Wang, Jeremy Wohlwend, and Tao Lei. Structured pruning of large language models. In *EMNLP*, 2020.

[Workshop, 2023] BigScience Workshop. BLOOM: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2023.

[Wu *et al.*, 2020] Yiquan Wu, Kun Kuang, Yating Zhang, Xiaozhong Liu, Changlong Sun, Jun Xiao, Yueting Zhuang,

Luo Si, and Fei Wu. De-biased court's view generation with causality. In *EMNLP*, pages 763–780, 2020.

[Wu *et al.*, 2023] Likang Wu, Zhi Zheng, Zhaopeng Qiu, et al. A survey on large language models for recommendation. *arXiv preprint arXiv:2305.19860*, 2023.

[Xia *et al.*, 2024] Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. Sheared LLaMA: Accelerating language model pre-training via structured pruning. In *ICLR*, 2024.

[Xiao *et al.*, 2023] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *ICML*, 2023.

[Yu *et al.*, 2022] Xin Yu, Thiago Serra, Srikumar Ramalingam, and Shandian Zhe. The combinatorial brain surgeon: Pruning weights that cancel one another in neural networks. In *ICML*, 2022.

[Zellers *et al.*, 2019] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *ACL*, 2019.

[Zhang *et al.*, 2022] Susan Zhang, Stephen Roller, Naman Goyal, et al. OPT: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

[Zhou *et al.*, 2022] Minxuan Zhou, Weihong Xu, Jaeyoung Kang, and Tajana Rosing. TransPIM: A memory-based acceleration via software-hardware co-design for transformer. In *HPCA*, 2022.

[Zhu *et al.*, 2015] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *ICCV*, 2015.