

EXPLORING FEDERATED PRUNING FOR LARGE LANGUAGE MODELS

Anonymous authors

Paper under double-blind review

ABSTRACT

LLM pruning has emerged as a promising technology for compressing LLMs, enabling their deployment on resource-limited devices. However, current methodologies typically require access to public calibration samples, which can be challenging to obtain in privacy-sensitive domains. To address this issue, we introduce **FedPrLLM**, a comprehensive federated pruning framework designed for the privacy-preserving compression of LLMs. In FedPrLLM, each client only needs to calculate a pruning mask matrix based on its local calibration data and share it with the server to prune the global model. This approach allows for collaborative pruning of the global model with the knowledge of each client while maintaining local data privacy. Additionally, we conduct extensive experiments to explore various possibilities within the FedPrLLM framework, including different comparison groups, pruning strategies, and the decision to scale weights. Our extensive evaluation reveals that one-shot pruning with layer comparison and no weight scaling is the optimal choice within the FedPrLLM framework. We hope our work will help guide future efforts in pruning LLMs in privacy-sensitive fields. Our code is available at <https://anonymous.4open.science/r/FedPrLLM-15594>.

1 INTRODUCTION

Large Language Models (LLMs) (Brown, 2020; Touvron et al., 2023a; Achiam et al., 2023) have revolutionized the field of natural language processing by demonstrating remarkable capabilities across various tasks. However, their increasing size leads to significant hardware requirements, limiting real-world deployment. To address this, research has focused on compact LLMs through compression techniques, such as *pruning* (Ma et al., 2023; Frantar & Alistarh, 2023; Sun et al., 2024), *knowledge distillation* (Gu et al., 2024; Xu et al., 2024b), *quantization* (Xiao et al., 2023; Shao et al., 2023), and *low-rank factorization* (Zhao et al., 2024; Saha et al., 2023). Among these, pruning has emerged as a promising method to reduce resource demands by selectively removing redundant parameters while preserving performance (Ma et al., 2023; Frantar & Alistarh, 2023). Typically, LLM pruning methods can be broadly classified into *structured pruning*, which removes entire substructures within LLMs, such as neurons (Ma et al., 2023; Li et al., 2023; Ashkboos et al., 2024), layers (Xia et al., 2024), or even entire transformer blocks (Gromov et al., 2025), and *unstructured pruning*, which removes individual weights from the model’s weight matrices based on certain criteria (Frantar & Alistarh, 2023; Sun et al., 2024; Zhang et al., 2024b; Yin et al., 2024; Xu et al., 2024a). This work focuses on unstructured pruning, as it tends to achieve higher compression rates and maintain better model performance compared to structured pruning (Frantar & Alistarh, 2023; He et al., 2024; Xia et al., 2024; Zhang et al., 2024b).

Despite advances in LLM unstructured pruning methods, these approaches usually rely on access to public calibration data to guide the pruning process (Frantar & Alistarh, 2023; Sun et al., 2024; Zhang et al., 2024b; Yin et al., 2024; Xu et al., 2024a). Specifically, they require calibration samples to evaluate the importance of the model weights in order to determine the pruning mask matrix for pruning models. However, in many real-world scenarios, such as healthcare, finance, and personalized services, the data used for pruning might be private and cannot be shared due to privacy regulations and concerns. Federated Learning (FL) (McMahan et al., 2017; Zhang et al., 2024a; Zeng et al., 2024; Guo et al., 2025b;a), which utilizes collaborative and decentralized training of models across multiple institutions without sharing personal data externally, offers a promising solution to this challenge.

Integrating FL with LLM pruning allows each client to calculate a local pruning mask matrix based on its private calibration data and share it with the server. The server then aggregates these mask matrices into an aggregated mask matrix and selects the top-k values (the most clients want to prune) to derive a final pruning mask matrix for pruning the global model. Despite its ability to protect data privacy, three unresolved challenges within this framework hinder practical deployment.

Challenge 1: How to compare parameters? When selecting the top-k values, a critical ambiguity arises: Should parameter importance be compared across the entire layer or within each respective row or column (corresponding to *layer*, *row*, and *column comparisons*, respectively)? Previous centralized LLM pruning work (Sun et al., 2024) has highlighted the importance of using a proper comparison group for pruning LLMs, yet no study explores this in federated scenarios.

Challenge 2: To scale or not scale for retained parameters. Beyond simply determining which parameters to prune via majority voting (i.e., selecting top-k values), the FL aggregated mask matrix reveals a critical hidden signal: how strongly each parameter is disfavored across clients. Consider two surviving parameters - one narrowly retained (pruned by 10/100 clients) and another unanimously preserved (pruned by 0/100 clients). Traditional pruning treats both equally, maintaining their original magnitudes despite their differing consensus levels. However, this ignores a critical insight: the former parameter, though retained, exhibits weaker consensus across clients. This observation raises a fundamental question: Rather than simply employing binary masking, could we leverage the FL aggregated mask matrix to guide continuous weight adjustment, where retained parameters are scaled down proportionally based on their pruning frequency?

Challenge 3: Is iterative pruning worth the cost? LLM pruning is typically performed *layer-by-layer* recursively to avoid error accumulation (Frantar & Alistarh, 2023; Sun et al., 2024; Zhang et al., 2024b). As a result, in FL, this necessitates either *one-shot pruning* (clients compute all layer mask matrices and share them with the server in one go) or *iterative pruning* (clients send the mask matrices to the server layer by layer in an iterative manner). While iterative pruning allows for refining the local model promptly, it incurs prohibitive communication costs for deep LLMs. This raises an unstudied question: Does iteratively refining the local model improve accuracy enough to justify its massive communication overhead?

To address these challenges, we formalize the first systematic study on federated LLM pruning and empirically evaluate three core design choices through a unified **FedPrLLM** framework (Figure 1):

Q1. Comparison Group: Which comparison group is more effective: *layer*, *row*, or *column*?

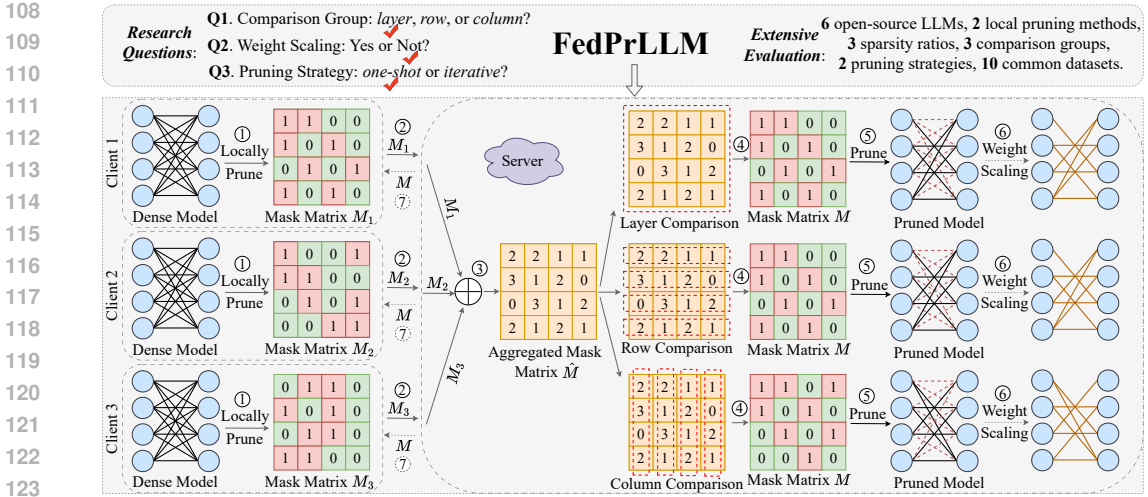
Q2. Weight Scaling: Should we scale the model weights of the retained parameters?

Q3. Pruning Strategy: Does iterative pruning outperform one-shot pruning?

We dedicated thousands of GPU hours to benchmark federated pruning for LLMs, conducting extensive experiments across **6** open-source LLMs, **2** local pruning methods, **3** sparsity ratios, **3** comparison groups, **2** pruning strategies on **10** common datasets. From these efforts, we have developed a practical list of key insights for federated pruning of LLMs:

- 1). Layer comparison is simple yet effective.** Among the three comparison groups—*layer*, *row*, and *column comparisons*—layer comparison stands out as the simplest and most effective method, regardless of the local pruning method’s comparison group.
- 2). Scaling weights performs worse than expected.** Though the FL aggregated mask matrix, which reveals how strongly each parameter is disfavored across clients, could be used to scale the retained parameters for continuous weight adjustment, its performance is inferior to that of not scaling them.
- 3). Iterative pruning offers no benefit.** While iterative pruning allows for prompt refinement of the local model, it incurs significant communication overhead, and its performance is comparable to that of one-shot pruning, offering no additional advantages.

We hope our findings will help guide future efforts in federated pruning for LLMs and inform best practices for deploying LLMs under federated scenarios in real-world applications. We summarize our contributions as follows:



124
125
126
127
128
129
130
131
132
133
134
135
136
137

Figure 1: *Top*). Research questions alongside the corresponding findings and experimental scenarios. *Bottom*). The FedPrLLM framework. ① Each client calculates a pruning mask matrix M_i using its calibration dataset \mathcal{D}_i . ② Clients send the mask matrices M_i to the server. ③ The server aggregates these mask matrices M_i to obtain an aggregated mask matrix $\hat{M} = \sum_{i=1}^m M_i$. ④ Top- k values are selected from the aggregated mask matrix \hat{W} to derive the final mask matrix M . ⑤ Prune the global model W using the mask matrix M as follows: $\hat{W} = W \odot (1 - M)$, where \odot denotes element-wise multiplication. ⑥ Scale the model weights of the retained parameters using the aggregated mask matrix \hat{M} as follows: $\hat{W} \odot \frac{(m - \hat{M})}{m}$ (if needed). ⑦ The server broadcasts the mask matrix M to each client (for iterative pruning). The dashed arrow indicates that this operation is optional; step ⑥ is used for weight scaling, while ⑦ is used for iterative pruning. Note that this visualization is primarily for one-shot pruning, which requires only one communication round. For iterative pruning, multiple communication rounds will occur between steps ② and ⑦, and the layer index is omitted here.

- We introduce **FedPrLLM**, a comprehensive federated pruning framework designed for the privacy-preserving compression of LLMs, which incorporates various possibilities for integrating FL with LLM pruning.
- We conduct an extensive evaluation of FedPrLLM, providing practical insights into effective federated pruning techniques for LLMs, based on thousands of GPU hours invested in multiple open-source LLMs, various sparsity ratios, comparison groups, and datasets.
- We identify that layer comparison is simple yet effective, scaling weights offers no benefits and may worsen performance, and that one-shot pruning is as effective as iterative pruning while reducing communication costs.

149 2 PRELIMINARIES

150
151
152
153
154
155
156
157
158
159
160

In this section, we review some concepts related to LLM pruning. LLM pruning can be broadly classified into *structured pruning* (Ma et al., 2023; Li et al., 2023; Ashkboos et al., 2024; Xia et al., 2024; Gromov et al., 2025) and *unstructured pruning* (Frantar & Alistarh, 2023; Sun et al., 2024; Zhang et al., 2024b; Yin et al., 2024; Xu et al., 2024a), and in this work, we focus on the latter. Unstructured pruning involves removing individual weights from the model’s weight matrices based on certain criteria while maintaining its performance as much as possible (Frantar & Alistarh, 2023; Sun et al., 2024; Zhang et al., 2024b; Yin et al., 2024; Xu et al., 2024a). It is usually achieved by minimizing the discrepancy square error between the dense and pruned model *layer-by-layer* recursively. Specifically, for an uncompressed linear layer with weights $W_l \in \mathbb{R}^{d \times r}$, the objective for unstructured pruning can usually be formulated as:

$$161 \quad \arg \min_{M_l} \|W_l X_l - (W_l \odot (1 - M_l)) X_l\|_2^2 \quad \text{s.t.} \quad \|M_l\|_0 \geq k, \quad (1)$$

where \mathbf{X}_l is the input to l -th linear layer (also referred to as calibration data), $\mathbf{M}_l \in \{0, 1\}^{d \times r}$ is the pruning mask matrix we aim to derive, \odot denotes element-wise multiplication, $\|\cdot\|_0$ is the l_0 -norm (e.g., the number of non-zero elements), and k represents the number of pruned weights determined by the pruning ratio.

The differences between previous pruning methods primarily lie in the design of the pruning metrics and the comparison groups used to derive the pruning mask matrix (Frantar & Alistarh, 2023; Sun et al., 2024; Zhang et al., 2024b). Pruning metrics refer to how the importance of each model weight is identified, while comparison groups denote the selection of groups for comparing these weights, including *layer comparison*, *row comparison*, and *column comparison*. For example, SparseGPT (Frantar & Alistarh, 2023) utilizes the Hessian Matrix inverse, i.e., $\left[\frac{|\mathbf{W}|^2}{\text{diag}(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1}} \right]_{ij}$, as the pruning metric, employing layer comparison to determine the pruning mask matrix for pruning, along with subsequent weight scaling. Wanda (Sun et al., 2024) adopts the magnitudes of model weights multiplied by the corresponding input activations, i.e., $|\mathbf{W}_{ij}| \cdot \|\mathbf{X}_j\|_2$, as the pruning metric and chooses row comparison.

3 FEDERATED PRUNING FOR LLMs

3.1 PROBLEM FORMULATION

In the federated pruning scenario for LLMs, multiple clients aim to collaboratively prune an LLM while ensuring that their local calibration data remains private. Formally, let \mathbf{W} represent the model parameters of the LLM that we aim to prune. Each client i possesses a private calibration dataset denoted as \mathcal{D}_i , which is used for calculating the pruning mask matrices during the local pruning process. These mask matrices are then shared with the server to prune the LLM.

3.2 FEDPRLLM

In this section, we first introduce the overall workflow of the comprehensive **FedPrLLM** framework, as illustrated at the bottom of Figure 1, and then discuss the various possibilities within it. Specifically, during local pruning, each client calculates a pruning mask matrix $\mathbf{M}_i \in \{0, 1\}^{|\mathbf{W}_i|}$ using its calibration dataset \mathcal{D}_i (step ①). This mask matrix determines which weights are pruned ($\mathbf{M}_{ij} = 1$) and which are retained ($\mathbf{M}_{ij} = 0$). The decision on which weights to prune or retain is based on an importance criterion derived from the calibration data, such as the magnitudes of model weights multiplied by the corresponding input activations used in Wanda (Sun et al., 2024) or other pruning methods.

After calculating the pruning mask matrix, each client i shares only the mask matrix \mathbf{M}_i with the central server (step ②). This approach ensures that no local model parameters or private calibration data are transmitted, thereby minimizing communication overhead and preserving data privacy. Upon receiving the pruning mask matrices \mathbf{M}_i from all clients, the server sums them to obtain an aggregated mask matrix $\hat{\mathbf{M}} = \sum_{i=1}^m \mathbf{M}_i$ (step ③) and then selects the top- k values to create the final mask matrix \mathbf{M} (step ④)¹ for pruning the global model (step ⑤). In the following, we will discuss various possibilities within the FedPrLLM framework, including different comparison groups, the decision to perform weight scaling, and the choice between one-shot and iterative pruning.

3.2.1 COMPARISON GROUP

When selecting the top- k values from the aggregated mask matrix $\hat{\mathbf{M}}$ to derive the final pruning mask matrix \mathbf{M} , three comparison groups can be considered (step ④): *layer comparison*, *row comparison*, and *column comparison*. In layer comparison, the comparison group consists of all elements within a layer, allowing us to choose the top- k values across the entire layer. Conversely, in row (or column) comparison, the comparison group is defined by each individual row (or column), enabling the selection of the top- k values within each respective row (or column). The visualization of these comparison groups is shown in Figure 1. Thus, given that multiple comparison groups could be chosen, *which comparison group is more effective for federated pruning of LLMs?*

¹The rationale behind such voting mechanism is shown in Section A in Appendix.

3.2.2 WEIGHT SCALING

After obtaining the final mask matrix \mathbf{M} , it can be used to effectively prune the dense model \mathbf{W} using $\mathbf{W} \odot (1 - \mathbf{M})$, where \odot denotes element-wise multiplication (step ⑤). This operation removes the weights corresponding to the masked parameters (i.e., $\mathbf{M}_{ij} = 1$), resulting in a sparser model $\hat{\mathbf{W}}$.

Then, beyond merely determining which parameters to prune via majority voting (i.e., selecting top-k values), the aggregated mask matrix $\hat{\mathbf{M}}$ reveals a critical hidden signal: how strongly each parameter is disfavored across clients. Consider two surviving parameters - one narrowly retained (pruned by 10/100 clients) and another unanimously preserved (pruned by 0/100 clients). Traditional pruning treats both equally, maintaining their original magnitudes despite their differing consensus levels. However, this ignores a critical insight: the former parameter, though retained, exhibits weaker consensus across clients. To this end, the aggregated mask matrix $\hat{\mathbf{M}}$ could be further applied to scale down the retained parameters using the formula $\hat{\mathbf{W}} \odot \frac{(m - \hat{\mathbf{M}})}{m}$ (step ⑥, if needed). This approach corresponds to locally pruning the model and then sharing the pruned model with the server, which aggregates them using the FedAvg algorithm (McMahan et al., 2017). However, *will the weight scaling improve the performance of federated pruning for LLMs?*

3.2.3 ONE-SHOT VS. ITERATIVE PRUNING

Since LLMs are usually pruned *layer-by-layer* recursively (Frantar & Alistarh, 2023; Sun et al., 2024; Zhang et al., 2024b), federated pruning for LLMs can be naturally categorized into two types: *one-shot pruning* and *iterative pruning*. In one-shot pruning, each client calculates the pruning mask matrices for all layers and then sends them to the server, resulting in only one communication round. In contrast, iterative pruning involves sending the pruning mask matrices to the server layer by layer. Specifically, after calculating the pruning mask matrix for one layer, it is uploaded to the server for aggregation. The server then combines these matrices into a global mask matrix for pruning the model at that layer and broadcasts the global mask matrix back to each client for local pruning of that layer (step ⑦, the layer index is omitted here). This process is carried out layer by layer and involves multiple communication rounds, resulting in higher communication costs compared to one-shot pruning. Therefore, given the significant communication costs associated with iterative pruning, *will iterative pruning outperform one-shot pruning?*

One-shot and iterative pruning differ because, when calculating the pruning mask matrix for layer $l + 1$ locally, the calibration data \mathbf{X}_{l+1} is derived from the output of layer l , which has already been pruned. Since the weights of the local pruned model for layer l vary between using \mathbf{M}_i (one-shot pruning) and \mathbf{M} (iterative pruning), this leads to different outputs for layer l and, consequently, varying calibration data \mathbf{X}_{l+1} , resulting in distinct pruning mask matrices for layer $l + 1$.

4 EXPERIMENTS

Our experiments are designed to answer the following research questions that are important for the practical pruning of LLMs under a federated scenario.

- **Q1.** Which comparison group is more effective: *layer*, *row*, or *column*?
- **Q2.** Should we scale the model weights of the retained parameters?
- **Q3.** Does iterative pruning outperform one-shot pruning?

4.1 EXPERIMENTAL SETUP

We implement FedPrLLM in PyTorch (Paszke et al., 2019) and use the Hugging Face Transformers library (Wolf et al., 2019) to handle models and datasets. We evaluate the FedPrLLM on the three most widely adopted LLM model families: LLaMA 7B/13B/30B (Touvron et al., 2023a), LLaMA-2 7B/13B (Touvron et al., 2023b) and LLaMA-3 8B (Meta, 2024). For each model under consideration, we focus on pruning the linear layers (skipping the first embedding layer and the final classification head), which account for around 99% of the total LLM parameters. We employ unstructured sparsity and impose a uniform sparsity ratio for all linear layers.

For the calibration data, following (Frantar & Alistarh, 2023; Sun et al., 2024; Xu et al., 2024a; Zhang et al., 2024b), we use 128 samples from the C4 dataset (Raffel et al., 2020), with each sample containing 2048 tokens. For FedPrLLM, we set the number of clients to 64, resulting in each client having only 2 calibration samples. For each client, we adopt Wanda (Sun et al., 2024) and SparseGPT (Frantar & Alistarh, 2023) to perform local pruning and calculate the pruning mask matrix.

Apart from the proposed FedPrLLM framework, we further implement two baselines for comparison: (1) **Local-only**, where each client prunes the model locally using its private calibration data, and (2) **Centralized**, where the server prunes the model with all calibration data, which could be considered as an upper bound for the pruning performance under FL setting.

Following previous works on LLM compression (Frantar & Alistarh, 2023; Xu et al., 2024a; Zhang et al., 2024b), we measure the performance of pruned models in language modeling and evaluate their perplexity on the held-out WikiText2 (Merity et al., 2017) validation set, C4 (Raffel et al., 2020) validation data, and PTB (Marcus et al., 1994). For further evaluation, we also assess the pruned models on seven zero-shot tasks from lm-evaluation-harness²: BoolQ (Clark et al., 2019), RTE (Wang et al., 2018), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC Easy and Challenge (Clark et al., 2018), and OpenbookQA (Mihaylov et al., 2018). The evaluation metric is accuracy.

4.2 MAIN RESULTS

To answer the research questions above, we conducted extensive experiments to evaluate FedPrLLM along with two baselines across **6** open-source LLMs, **2** local pruning methods, **3** sparsity ratios, **3** comparison groups, **2** pruning strategies on **10** common datasets. The experimental results using Wanda as the local pruning method for the 50% sparsity ratio on the WikiText2 dataset are shown in Table 1, while results for higher sparsity ratios (e.g., 60% and 70%) and other datasets (e.g., C4 and PTB) are shown in Tables 4, 5, and 6 in Appendix. More results using SparseGPT as the local pruning method and evaluation on the zero-shot tasks are shown in Tables 8, 9, 10, and 11 in Appendix.

Table 1: WikiText2 perplexity of pruned LLMs under 50% sparsity ratio using Wanda as the local pruning method.

Method	Compar. Group	Prune Stra.	Weight Scaling	LLaMA			LLaMA-2		LLaMA-3
				7B	13B	30B	7B	13B	8B
Dense	-	-	-	5.67	5.09	4.10	5.11	4.57	7.46
Centralized	-	-	-	7.25	6.15	5.24	6.46	5.58	11.00
Local-only	-	-	-	7.44	6.33	5.34	6.63	5.72	11.39
FedPrLLM	Layer	One-shot	X	7.32	6.19	5.24	6.48	5.61	11.02
	Row	One-shot	X	7.30	6.20	5.25	6.48	5.61	11.02
	Column	One-shot	X	1524.28	9282.09	501.88	20528.41	5309.48	311468.53
	Layer	Iterative	X	7.30	6.19	5.24	6.48	5.62	11.12
	Row	Iterative	X	7.30	6.20	5.24	6.48	5.61	11.11
	Column	Iterative	X	1822.89	6884.15	996.57	77245.84	5430.81	189134.78
	Layer	One-shot	✓	7.48	6.36	5.35	6.67	5.75	11.75
	Row	One-shot	✓	7.47	6.36	5.35	6.67	5.75	11.75
	Column	One-shot	✓	1708.41	10819.42	824.50	18084.02	5914.91	276031.34
	Layer	Iterative	✓	7.46	6.35	5.34	6.67	5.75	11.86
	Row	Iterative	✓	7.46	6.35	5.34	6.67	5.74	11.87
	Column	Iterative	✓	1985.40	6692.91	939.62	66911.49	5268.71	41996.95

4.2.1 WHICH COMPARISON GROUP IS MORE EFFECTIVE?

As discussed above, various comparison groups can be used to select top-k values from the aggregated mask matrix to derive the final mask matrix for pruning the global model, including *layer comparison*, *row comparison*, and *column comparison*. Thus, which comparison group is the most effective?

²<https://github.com/EleutherAI/lm-evaluation-harness>

According to the results in Table 1, we observe that layer comparison and row comparison achieve comparable performance, both significantly surpassing column comparison. Results on higher sparsity ratios and other datasets (Tables 4, 5, and 6 in Appendix), using another local pruning method (Table 8 in Appendix), and results on zero-shot tasks (Table 10 in Appendix) show a similar phenomenon. To investigate why column comparison performs much worse than the others, we noted that the local pruning methods we used adopts row comparison, meaning the local pruning mask matrix M_i derived from each client is based on row comparison. We hypothesize that this is the reason for the poorer performance of column comparison, as the comparison group used in FedPrLLM conflicts with that of the local pruning method.

Table 2: WikiText2 perplexity of pruned LLMs under 50% sparsity ratio when changing the comparison group for the local pruning method (i.e., Wanda). FedPrLLM adopts one-shot pruning and no weight scaling.

Local Compar.		Compar.	LLaMA			LLaMA-2		LLaMA-3
Group	Method	Group	7B	13B	30B	7B	13B	8B
Layer	Centralized	-	7.94	6.57	5.47	7.38	5.92	12.04
	Local-only	-	8.16	6.74	5.58	7.56	6.06	12.43
	FedPrLLM	Layer	7.98	6.60	5.48	7.38	5.95	12.09
		Row	31.85	10.08	11.33	39.07	124.08	17.51
		Column	1749.59	10183.32	541.62	25258.16	5503.91	336255.96
	Column	Centralized	-	8.86	7.68	5.67	10.41	6.38
Local-only		-	8.86	7.68	5.67	10.41	6.38	83.67
FedPrLLM		Layer	8.86	7.68	5.67	10.41	6.38	83.67
		Row	138.54	100.80	49.17	764.32	2580.88	400.95
		Column	8.86	7.68	5.67	10.41	6.38	83.67

To validate this, we further change the comparison group in the local pruning method (i.e., Wanda (Sun et al., 2024) and SparseGPT (Frantar & Alistarh, 2023)) to layer comparison and column comparison to evaluate the performance of the FedPrLLM framework with one-shot pruning and no weight scaling. The results on WikiText2 are shown in Table 2, while results for other datasets are presented in Table 7 in Appendix. More results using SparseGPT as the local pruning method and results on the zero-shot tasks are shown in Tables 9 and 11 in Appendix. From these results, we see that when the comparison group in the local pruning method is changed to layer comparison, only the layer comparison used in FedPrLLM performs well, while row comparison performs poorly and column comparison performs even worse. Similarly, when the local pruning method’s comparison group is changed to column comparison, only the layer and column comparisons perform normally, while row comparison performance is poor. Note that when the comparison group in the local pruning method is changed to column comparison, it degrades to the magnitude-based pruning method, rendering the performance irrelevant to calibration samples, which results in the performance of Centralized and Local-only being the same (Sun et al., 2024). These results demonstrate our hypothesis that the conflict between the local and server comparison groups leads to worse performance, while the layer comparison used in FedPrLLM consistently achieves good results, regardless of the comparison group used for the local pruning method. The reason for this phenomenon may be due to the mismatch between the local and server comparison groups, which renders the aggregated mask matrix “meaningless”. We know that the aggregated mask matrix can be considered a “weight importance matrix” for conducting pruning on the server side. Note that these importance values are only meaningful under the local comparison group and will be meaningless under a mismatched comparison group. Therefore, when the comparison group used on the server mismatches the local group (e.g., local-row and server-column), the aggregated mask matrix will be meaningless and cannot be used to determine which weights are important, leading to poor pruning results. However, the layer comparison used on the server can avoid this issue since the comparisons within the whole layer will also take the local comparison group into consideration. Thus, regardless of the local comparison group used on the client side, the layer comparison used on the server can achieve good results. Therefore, we conclude that:

Takeaway 1: Layer comparison is simple yet effective.

4.2.2 SHOULD WE SCALE THE MODEL WEIGHTS OF THE RETAINED PARAMETERS?

The aggregated mask matrix \hat{M} indicates the number of clients that wish to prune a parameter, which allows it to be used for scaling the model weights of the retained parameters to $\frac{(m-\hat{M})}{m}$. This approach corresponds to locally pruning the model and then sharing the pruned model with the server, which aggregates them using the FedAvg algorithm (McMahan et al., 2017). However, will weight scaling be beneficial for the federated pruning of LLMs?

From the results in Table 1, we observe that the performance with weight scaling is worse than that without weight scaling across all comparison groups and pruning strategies. Results on higher sparsity ratios and more datasets (Tables 4, 5, and 6 in Appendix), using another local pruning method (Table 8 in Appendix), and results on zero-shot tasks (Table 10 in Appendix) show a similar phenomenon. It indicates that scaling weights offers no benefit and may even worsen performance. This may be due to the fact that locally pruned models do not perform well, and applying the FedAvg algorithm (McMahan et al., 2017) to aggregate these pruned model weights leads to subpar performance. Therefore, we conclude that:

Takeaway 2: Scaling weights performs worse than expected.

4.2.3 DOES ITERATIVE PRUNING OUTPERFORM ONE-SHOT PRUNING?

Since LLMs are usually pruned *layer-by-layer* recursively (Frantar & Alistarh, 2023; Sun et al., 2024; Zhang et al., 2024b), federated pruning for LLMs can be naturally categorized into two types: *one-shot pruning* and *iterative pruning*. Given the significant communication costs associated with iterative pruning, will it outperform one-shot pruning?

Table 3: Communication cost for one-shot and iterative pruning.

	LLaMA-7B	LLaMA-13B	LLaMA-30B	LLaMA-2-7B	LLaMA-2-13B	LLaMA-3-8B
one-shot pruning	6.476B	12.688B	32.102B	6.476B	12.688B	6.979B
iterative pruning	12.952B	25.376B	64.204B	12.952B	25.376B	13.958B

The comparison results are provided in Table 1, More results on higher sparsity ratios and other datasets are shown in Tables 4, 5, and 6 in Appendix. Results using another local pruning method are shown in Table 8 in Appendix, and results on zero-shot tasks are shown in Table 10 in Appendix. These results indicate that the performance of iterative pruning and one-shot pruning is comparable, regardless of the comparison groups and pruning strategies. However, since iterative pruning introduces significant communication costs (Table 3) without any performance improvement, we conclude that:

Takeaway 3: Iterative pruning offers no benefit.

4.3 SENSITIVITY ANALYSIS

In this section, we conduct sensitivity analyses on the number of clients and calibration samples in FedPrLLM to better understand its effectiveness in pruning LLMs within a federated scenario. We utilize Wanda as the local pruning method and use FedPrLLM, which employs layer comparison, one-shot pruning, and no weight scaling, to conduct the analysis under a 50% sparsity ratio.

4.3.1 IMPACT OF CLIENT NUMBERS

It is worth noting that the number of clients influences the performance of FL algorithms (Guo et al., 2025b;c). In this section, we investigate the effect of client numbers on the federated pruning of LLMs. We use a total of 128 calibration samples and vary the number of clients from 64 to 2, resulting in an increase in the calibration samples allocated to each client. Specifically, when the number of clients is 64, each client has only 2 calibration samples; when the number of clients is reduced to 2, each client has 64 calibration samples. The experimental results are shown in Figure

2. From this figure, we observe that FedPrLLM consistently outperforms Local-only pruning across various numbers of clients, demonstrating the effectiveness of the federated pruning algorithm.

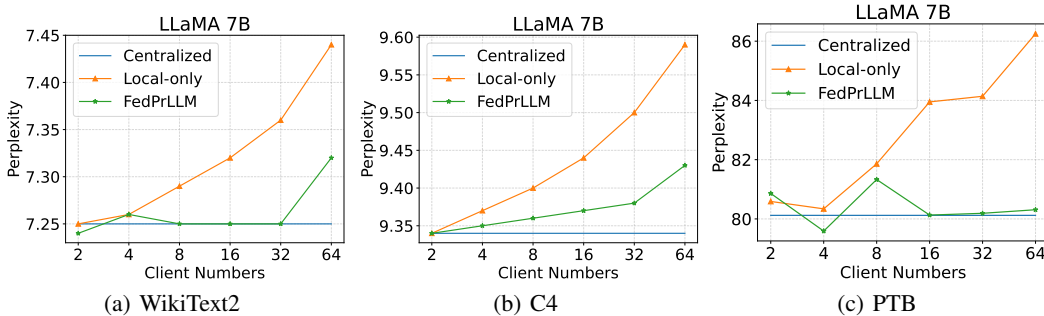


Figure 2: The effect of different client numbers on federated pruning LLMs.

4.3.2 IMPACT OF THE NUMBER OF CALIBRATION SAMPLES

We further investigate the impact of pruning LLMs in a federated scenario with varying numbers of calibration samples, as shown in Figure 3. Specifically, we change the total number of calibration samples from 128 to 4 while keeping the number of clients equal to half of that. As shown in Figure 3, we observe that with different numbers of calibration samples, FedPrLLM consistently outperforms Local-only pruning, which again shows the effectiveness of the federated pruning method.

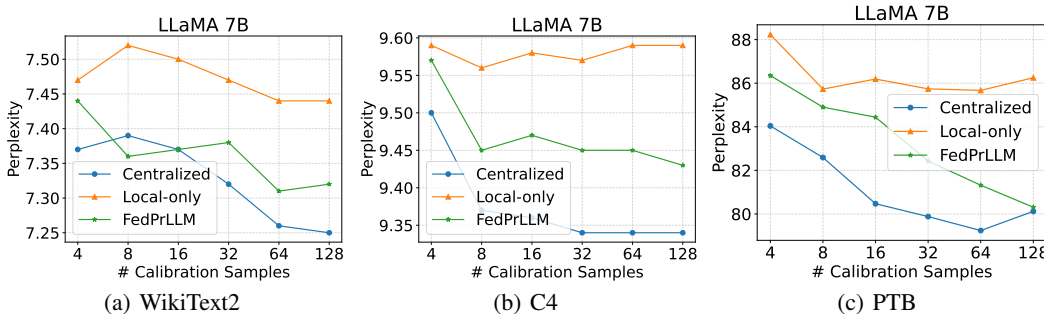


Figure 3: The effect of the number of calibration samples on federated pruning LLMs.

5 RELATED WORK

There is one work that attempts to conduct LLM pruning in an FL scenario (Bai et al., 2024). Specifically, it enables clients to locally prune their models based on private data and send the pruned models to the server for aggregation. The server averages the pruned models using the FedAvg algorithm (McMahan et al., 2017) and prunes the model to satisfy the predefined sparsity rate based on an aggregated mask matrix. This method can be viewed as a specific case within our FedPrLLM framework, i.e., iterative pruning with weight scaling. However, our extensive evaluations reveal that this approach is not optimal.

6 CONCLUSION

In this work, we introduce **FedPrLLM**, a comprehensive federated pruning framework designed for the privacy-preserving compression of LLMs, incorporating various possibilities for integrating FL with LLM pruning. To identify the optimal operation within this framework, we invested thousands of GPU hours exploring these possibilities, including different comparison groups, pruning strategies, and the decision to scale weights. Our extensive evaluation reveals that one-shot pruning with layer comparison and no weight scaling is the optimal choice within the FedPrLLM framework. We hope our work will help guide future efforts in pruning LLMs in privacy-sensitive fields.

REFERENCES

- 486
487
488 Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Ale-
489 man, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical
490 report. *arXiv preprint arXiv:2303.08774*, 2023.
- 491 Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and
492 James Hensman. SliceGPT: Compress large language models by deleting rows and columns.
493 In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=vXxardq6db>.
- 494
495 Guangji Bai, Yijiang Li, Zilinghan Li, Liang Zhao, and Kibaek Kim. Fedspallm: Federated pruning
496 of large language models. *arXiv preprint arXiv:2410.14852*, 2024.
- 497
498 Tom B Brown. Language models are few-shot learners. *arXiv preprint ArXiv:2005.14165*, 2020.
- 499
500 Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina
501 Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings*
502 *of the 2019 Conference of the North American Chapter of the Association for Computational*
503 *Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936,
504 2019.
- 505
506 Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and
507 Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge.
508 *arXiv preprint arXiv:1803.05457*, 2018.
- 509
510 Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in
511 one-shot. In *International Conference on Machine Learning*, pp. 10323–10337. PMLR, 2023.
- 512
513 Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Dan Roberts. The
514 unreasonable ineffectiveness of the deeper layers. In *The Thirteenth International Confer-*
515 *ence on Learning Representations*, 2025. URL [https://openreview.net/forum?id=](https://openreview.net/forum?id=ngmEcEer8a)
516 [ngmEcEer8a](https://openreview.net/forum?id=ngmEcEer8a).
- 517
518 Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large
519 language models. In *The Twelfth International Conference on Learning Representations*, 2024.
520 URL <https://openreview.net/forum?id=5h0qf7IBZZ>.
- 521
522 Pengxin Guo, Runxi Wang, Shuang Zeng, Jinjing Zhu, Haoning Jiang, Yanran Wang, Yuyin Zhou,
523 Feifei Wang, Hui Xiong, and Liangqiong Qu. Exploring the vulnerabilities of federated learning:
524 A deep dive into gradient inversion attacks. *arXiv preprint arXiv:2503.11514*, 2025a.
- 525
526 Pengxin Guo, Shuang Zeng, Wenhao Chen, Xiaodan Zhang, Weihong Ren, Yuyin Zhou, and
527 Liangqiong Qu. A new federated learning framework against gradient inversion attacks. In *Pro-*
528 *ceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 16969–16977, 2025b.
- 529
530 Pengxin Guo, Shuang Zeng, Yanran Wang, Huijie Fan, Feifei Wang, and Liangqiong Qu. Selective
531 aggregation for low-rank adaptation in federated learning. In *The Thirteenth International Con-*
532 *ference on Learning Representations*, 2025c. URL [https://openreview.net/forum?](https://openreview.net/forum?id=iX3uESGdsO)
533 [id=iX3uESGdsO](https://openreview.net/forum?id=iX3uESGdsO).
- 534
535 Shwai He, Guoheng Sun, Zheyu Shen, and Ang Li. What matters in transformers? not all attention
536 is needed. *arXiv preprint arXiv:2406.15786*, 2024.
- 537
538 Yixiao Li, Yifan Yu, Qingru Zhang, Chen Liang, Pengcheng He, Weizhu Chen, and Tuo Zhao.
539 Lospars: Structured compression of large language models based on low-rank and sparse approx-
imation. In *International Conference on Machine Learning*, pp. 20336–20350. PMLR, 2023.
- Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large
language models. *Advances in neural information processing systems*, 36:21702–21720, 2023.
- Mitch Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson,
Karen Katz, and Britta Schasberger. The penn treebank: Annotating predicate argument structure.
In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey,*
March 8-11, 1994, 1994.

- 540 Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas.
541 Communication-efficient learning of deep networks from decentralized data. In *Artificial intelli-*
542 *gence and statistics*, pp. 1273–1282. PMLR, 2017.
- 543
544 Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mix-
545 ture models. In *International Conference on Learning Representations*, 2017. URL <https://openreview.net/forum?id=Byj72udxe>.
546
- 547 AI Meta. Introducing meta llama 3: The most capable openly available llm to date. *Meta AI*, 2024.
548
- 549 Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct elec-
550 tricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference*
551 *on Empirical Methods in Natural Language Processing*, pp. 2381–2391, 2018.
- 552
553 Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor
554 Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward
555 Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner,
556 Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance
557 deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox,
558 and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Cur-
559 ran Associates, Inc., 2019. URL [https://proceedings.neurips.cc/paper_files/](https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf)
560 [paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf).
- 561 Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi
562 Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text
563 transformer. *Journal of machine learning research*, 21(140):1–67, 2020.
- 564
565 Rajarshi Saha, Varun Srivastava, and Mert Pilanci. Matrix compression via randomized low rank
566 and low precision factorization. *Advances in Neural Information Processing Systems*, 36, 2023.
- 567
568 Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adver-
569 sarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.
- 570
571 Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang,
572 Peng Gao, Yu Qiao, and Ping Luo. Omniquant: Omnidirectionally calibrated quantization for
573 large language models. *arXiv preprint arXiv:2308.13137*, 2023.
- 574
575 Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach
576 for large language models. In *The Twelfth International Conference on Learning Representations*,
577 2024. URL <https://openreview.net/forum?id=PxoFut3dWW>.
- 578
579 Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée
580 Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and
581 efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.
- 582
583 Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Niko-
584 lay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open founda-
585 tion and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.
- 586
587 Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman.
588 Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv*
589 *preprint arXiv:1804.07461*, 2018.
- 590
591 Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi,
592 Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface’s transformers:
593 State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.
- 594
595 Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. Sheared LLaMA: Accelerat-
596 ing language model pre-training via structured pruning. In *The Twelfth International Confer-*
597 *ence on Learning Representations*, 2024. URL [https://openreview.net/forum?id=](https://openreview.net/forum?id=09iOdaeOzp)
598 [09iOdaeOzp](https://openreview.net/forum?id=09iOdaeOzp).

- 594 Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant:
595 Accurate and efficient post-training quantization for large language models. In *International
596 Conference on Machine Learning*, pp. 38087–38099. PMLR, 2023.
597
- 598 Peng Xu, Wenqi Shao, Mengzhao Chen, Shitao Tang, Kaipeng Zhang, Peng Gao, Fengwei An,
599 Yu Qiao, and Ping Luo. BESA: Pruning large language models with blockwise parameter-efficient
600 sparsity allocation. In *The Twelfth International Conference on Learning Representations*, 2024a.
601 URL <https://openreview.net/forum?id=gC6JTEU3jl>.
- 602 Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng
603 Tao, and Tianyi Zhou. A survey on knowledge distillation of large language models. *arXiv
604 preprint arXiv:2402.13116*, 2024b.
- 605 Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Gen Li, AJAY KUMAR
606 JAISWAL, Mykola Pechenizkiy, Yi Liang, Michael Bendersky, Zhangyang Wang, and Shiwei
607 Liu. Outlier weighed layerwise sparsity (OWL): A missing secret sauce for pruning LLMs to
608 high sparsity. In *Forty-first International Conference on Machine Learning*, 2024. URL <https://openreview.net/forum?id=ahEm3l2P6w>.
- 609
610
- 611 Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a ma-
612 chine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association
613 for Computational Linguistics*, pp. 4791–4800, 2019.
- 614 Shuang Zeng, Pengxin Guo, Shuai Wang, Jianbo Wang, Yuyin Zhou, and Liangqiong Qu. Tackling
615 data heterogeneity in federated learning via loss decomposition. In *International Conference on
616 Medical Image Computing and Computer-Assisted Intervention*, pp. 707–717. Springer, 2024.
617
- 618 Junyuan Zhang, Shuang Zeng, Miao Zhang, Runxi Wang, Feifei Wang, Yuyin Zhou, Paul Pu Liang,
619 and Liangqiong Qu. Flhetbench: Benchmarking device and state heterogeneity in federated learn-
620 ing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*,
621 pp. 12098–12108, 2024a.
- 622 Yingtao Zhang, Haoli Bai, Haokun Lin, Jialin Zhao, Lu Hou, and Carlo Vittorio Cannistraci. Plug-
623 and-play: An efficient post-training pruning method for large language models. In *The Twelfth
624 International Conference on Learning Representations*, 2024b.
- 625 Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong
626 Tian. Galore: Memory-efficient llm training by gradient low-rank projection. In *International
627 Conference on Machine Learning*, pp. 61121–61143. PMLR, 2024.
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647

A THE RATIONALE OF VOTING MECHANISMS

In this section, we provide theoretical analysis to demonstrate the rationale behind the voting mechanism for deriving the final pruning mask on the server side. Let $\{\mathbf{M}_1, \dots, \mathbf{M}_m\}$ be m independent $d \times r$ binary mask matrices (with 50% sparsity) where each matrix satisfies:

$$\sum_{p=1}^d \sum_{q=1}^r \mathbf{M}_i[p, q] = \frac{dr}{2}. \quad (2)$$

The voting mechanism procedure produces \mathbf{M} via: 1) Element-wise sum: $\hat{\mathbf{M}} = \sum_{i=1}^m \mathbf{M}_i$; 2) Set the largest $\frac{dr}{2}$ entries in $\hat{\mathbf{M}}$ to 1, others to 0.

Let \mathbf{M}^* be the optimal mask defined by:

$$\mathbf{M}^*[p, q] = \mathbb{I}(p_{pq} \geq \tau^*), \quad (3)$$

where $p_{pq} = P(\mathbf{M}_i[p, q] = 1)$ and τ^* is chosen such that $\sum_{p,q} \mathbf{M}^*[p, q] = \frac{dr}{2}$.

Then, the error between \mathbf{M} and \mathbf{M}^* can be defined as:

$$\epsilon = \frac{1}{dr} \sum_{p=1}^d \sum_{q=1}^r \mathbb{I}(\mathbf{M}[p, q] \neq \mathbf{M}^*[p, q]). \quad (4)$$

There are two situations for $\mathbf{M}^*[p, q]$: 1 or 0.

Case 1: $\mathbf{M}^*[p, q] = 1$ (i.e., $p_{pq} \geq \tau^*$). In this case, $\mathbf{M}[p, q] = 0$ implies $\frac{\hat{\mathbf{M}}[p, q]}{m} < \tau^*$. Thus:

$$p_{pq} - \frac{\hat{\mathbf{M}}[p, q]}{m} > p_{pq} - \tau^* = \delta_{pq} \quad (\text{since } \delta_{pq} = |p_{pq} - \tau^*| = p_{pq} - \tau^*), \quad (5)$$

which simplifies to:

$$\left| \frac{\hat{\mathbf{M}}[p, q]}{m} - p_{pq} \right| > \delta_{pq} \quad (6)$$

Case 2: $\mathbf{M}^*[p, q] = 0$ (i.e., $p_{pq} < \tau^*$). In this case, $\mathbf{M}[p, q] = 1$ implies $\frac{\hat{\mathbf{M}}[p, q]}{m} \geq \tau^*$. Thus:

$$\frac{\hat{\mathbf{M}}[p, q]}{m} - p_{pq} \geq \tau^* - p_{pq} = \delta_{pq} \quad (\text{since } \delta_{pq} = \tau^* - p_{pq}), \quad (7)$$

which simplifies to:

$$\left| \frac{\hat{\mathbf{M}}[p, q]}{m} - p_{pq} \right| \geq \delta_{pq} \quad (8)$$

Let: **Event A:** $\mathbf{M}[p, q] \neq \mathbf{M}^*[p, q]$; **Event B:** $\left| \frac{\hat{\mathbf{M}}[p, q]}{m} - p_{pq} \right| \geq \frac{\delta_{pq}}{2}$. Then $A \subseteq B$, and we have:

$$P(\mathbf{M}[p, q] \neq \mathbf{M}^*[p, q]) \leq P\left(\left| \frac{\hat{\mathbf{M}}[p, q]}{m} - p_{pq} \right| \geq \frac{\delta_{pq}}{2}\right) \leq 2 \exp\left(-\frac{m\delta_{pq}^2}{2}\right), \quad (9)$$

This implies:

$$\mathbb{E}[\epsilon] \leq \frac{2}{dr} \sum_{p,q} \exp\left(-\frac{m\delta_{pq}^2}{2}\right). \quad (10)$$

This shows that the error between \mathbf{M} (which is obtained by voting) and \mathbf{M}^* is bounded by some value, which demonstrates the rationale behind the voting mechanism.

B ADDITIONAL EXPERIMENTAL RESULTS

B.1 MORE RESULTS UNDER HIGHER SPARSITY RATIOS

The experimental results using Wanda as the local pruning method for higher sparsity ratios (i.e., 60% and 70%) are shown in Tables 4, 5, and 6.

Table 4: WikiText2 perplexity of pruned LLaMA, LLaMA-2, and LLaMA-3 models using Wanda as the local pruning method.

Sparsity	Method	Compar.	Prune	Weight	LLaMA			LLaMA-2		LLaMA-3	
		Group	Stra.	Scaling	7B	13B	30B	7B	13B	8B	
0%	Dense	-	-	-	5.67	5.09	4.10	5.11	4.57	7.46	
	Centralized	-	-	-	7.25	6.15	5.24	6.46	5.58	11.00	
	Local-only	-	-	-	7.44	6.33	5.34	6.63	5.72	11.39	
	50%	FedPrLLM	Layer	One-shot	✗	7.32	6.19	5.24	6.48	5.61	11.02
			Row	One-shot	✗	7.30	6.20	5.25	6.48	5.61	11.02
			Column	One-shot	✗	1524.28	9282.09	501.88	20528.41	5309.48	311468.53
			Layer	Iterative	✗	7.30	6.19	5.24	6.48	5.62	11.12
			Row	Iterative	✗	7.30	6.20	5.24	6.48	5.61	11.11
			Column	Iterative	✗	1822.89	6884.15	996.57	77245.84	5430.81	189134.78
			Layer	One-shot	✓	7.48	6.36	5.35	6.67	5.75	11.75
			Row	One-shot	✓	7.47	6.36	5.35	6.67	5.75	11.75
			Column	One-shot	✓	1708.41	10819.42	824.5	18084.02	5914.91	276031.34
			Layer	Iterative	✓	7.46	6.35	5.34	6.67	5.75	11.86
Row	Iterative	✓	7.46	6.35	5.34	6.67	5.74	11.87			
Column	Iterative	✓	1985.40	6692.91	939.62	66911.49	5268.71	41996.95			
Centralized	-	-	-	10.71	8.74	6.55	10.03	7.92	25.81		
Local-only	-	-	-	11.70	9.38	6.96	10.84	8.55	27.47		
60%	FedPrLLM	Layer	One-shot	✗	10.76	8.80	6.65	10.08	8.01	25.48	
		Row	One-shot	✗	10.77	8.80	6.64	10.08	8.03	25.64	
		Column	One-shot	✗	2861.56	11190.34	1047.94	14737.65	5385.33	382319.37	
		Layer	Iterative	✗	10.87	8.88	6.65	10.17	8.05	26.21	
		Row	Iterative	✗	10.85	8.90	6.64	10.18	8.05	25.98	
		Column	Iterative	✗	3154.68	7824.46	2250.97	18849.20	6556.50	65475.84	
		Layer	One-shot	✓	12.14	9.77	7.10	11.53	8.98	30.34	
		Row	One-shot	✓	12.16	9.77	7.09	11.53	9.00	30.44	
		Column	One-shot	✓	3785.85	17163.16	1770.89	15180.33	5401.19	608169.33	
		Layer	Iterative	✓	12.27	9.85	7.12	11.90	9.07	30.94	
		Row	Iterative	✓	12.24	9.86	7.13	11.87	9.06	31.08	
		Column	Iterative	✓	2189.53	6032.71	2626.57	16081.73	6227.41	165510.73	
		Centralized	-	-	-	87.42	53.48	17.30	72.38	45.94	92.20
Local-only	-	-	-	104.15	67.13	23.29	80.39	51.79	108.35		
70%	FedPrLLM	Layer	One-shot	✗	83.12	55.92	18.73	70.92	44.98	102.88	
		Row	One-shot	✗	81.97	56.99	18.67	70.61	44.66	102.13	
		Column	One-shot	✗	17281.43	13045.16	2670.43	31238.51	12206.74	458666.00	
		Layer	Iterative	✗	89.25	55.48	18.65	79.27	45.89	100.37	
		Row	Iterative	✗	92.29	57.18	18.23	72.60	45.68	93.13	
		Column	Iterative	✗	19791.05	10323.63	3935.54	23090.20	7857.41	355916.56	
		Layer	One-shot	✓	136.50	94.90	31.62	93.89	64.34	123.92	
		Row	One-shot	✓	136.09	95.86	31.48	93.36	63.98	124.65	
		Column	One-shot	✓	20505.56	11695.06	3032.65	31485.38	10875.86	831352.18	
		Layer	Iterative	✓	174.95	102.78	31.12	94.49	62.07	116.97	
		Row	Iterative	✓	182.73	99.32	30.87	96.37	62.51	120.19	
		Column	Iterative	✓	8607.36	11707.00	3145.32	36254172.00	9604.48	1034635.56	

756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809

Table 5: C4 perplexity of pruned LLaMA, LLaMA-2, and LLaMA-3 models using Wanda as the local pruning method.

Sparsity	Method	Compar. Group	Prune Stra.	Weight Scaling	LLaMA			LLaMA-2		LLaMA-3		
					7B	13B	30B	7B	13B	8B		
0%	Dense	-	-	-	7.34	6.79	6.12	7.03	6.51	12.34		
	Centralized	-	-	-	9.34	8.14	7.28	8.94	8.03	18.38		
	Local-only	-	-	-	9.59	8.37	7.52	9.16	8.31	18.92		
50%	FedPrLLM	Layer	One-shot	✗	9.43	8.22	7.39	9.01	8.18	18.32		
		Row	One-shot	✗	9.43	8.22	7.39	9.01	8.19	18.32		
		Column	One-shot	✗	893.05	10616.94	612.27	9631.37	5075.92	200257.70		
		Layer	Iterative	✗	9.44	8.22	7.39	9.01	8.19	18.43		
		Row	Iterative	✗	9.44	8.22	7.39	9.02	8.18	18.38		
		Column	Iterative	✗	1050.26	8567.66	779.01	11658.80	4804.46	112192.42		
		Layer	One-shot	✓	9.64	8.40	7.57	9.21	8.39	19.45		
		Row	One-shot	✓	9.64	8.40	7.57	9.21	8.39	19.45		
		Column	One-shot	✓	887.34	13744.66	895.18	11440.51	5189.73	90476.94		
		Layer	Iterative	✓	9.64	8.41	7.57	9.22	8.39	19.58		
		Row	Iterative	✓	9.65	8.41	7.57	9.22	8.39	19.60		
		Column	Iterative	✓	1242.31	6860.69	724.28	10355.87	4657.88	44469.52		
			Centralized	-	-	-	13.72	11.22	9.16	13.64	11.39	43.02
			Local-only	-	-	-	14.69	11.91	9.58	14.68	12.17	45.25
		60%	FedPrLLM	Layer	One-shot	✗	13.80	11.23	9.29	13.77	11.40	42.61
				Row	One-shot	✗	15.26	12.24	9.79	15.60	12.75	50.37
Column	One-shot			✗	2149.09	11488.68	993.56	12252.16	4606.43	837570.62		
Layer	Iterative			✗	13.92	11.37	9.32	13.84	11.52	44.24		
Row	Iterative			✗	13.86	11.38	9.30	13.85	11.53	43.77		
Column	Iterative			✗	2981.52	10375.02	1752.73	16673.62	5289.35	62234.32		
Layer	One-shot			✓	15.24	12.24	9.80	15.61	12.74	50.28		
Row	One-shot			✓	15.26	12.24	9.79	15.60	12.75	50.37		
Column	One-shot			✓	3336.72	19430.46	1520.32	14613.11	4547.54	622715.25		
Layer	Iterative			✓	15.46	12.54	9.86	16.15	13.01	51.47		
Row	Iterative			✓	15.42	12.54	9.87	16.10	13.01	51.48		
Column	Iterative			✓	1825.82	6669.63	1865.50	16167.12	5057.57	145341.28		
	Centralized	-	-	-	85.84	53.35	18.80	84.16	58.56	136.66		
	Local-only	-	-	-	96.47	63.61	22.48	82.96	67.09	161.86		
70%	FedPrLLM	Layer	One-shot	✗	81.95	52.55	19.24	81.40	59.87	158.08		
		Row	One-shot	✗	82.02	53.51	19.22	81.59	59.97	157.87		
		Column	One-shot	✗	15276.62	14041.01	2059.83	39339.21	11306.11	398674.93		
		Layer	Iterative	✗	83.52	57.22	19.15	92.51	60.46	162.29		
		Row	Iterative	✗	86.77	55.98	19.20	84.99	60.86	144.71		
		Column	Iterative	✗	18149.76	13537.18	2874.83	21704.32	7166.78	346598.5		
		Layer	One-shot	✓	116.61	77.99	26.30	104.86	79.82	184.11		
		Row	One-shot	✓	117.29	78.84	26.29	104.51	79.76	184.11		
		Column	One-shot	✓	19380.0	10934.98	2336.68	32034.07	11360.57	345798.53		
		Layer	Iterative	✓	142.08	85.91	27.17	103.02	79.25	177.97		
		Row	Iterative	✓	145.15	84.68	27.00	102.61	79.41	182.36		
		Column	Iterative	✓	7664.62	15985.50	2685.76	27805842.0	8041.09	1031318.56		

Table 6: PTB perplexity of pruned LLaMA, LLaMA-2, and LLaMA-3 models using Wanda as the local pruning method.

Sparsity	Method	Compar. Group	Prune Stra.	Weight Scaling	LLaMA			LLaMA-2		LLaMA-3		
					7B	13B	30B	7B	13B	8B		
0%	Dense	-	-	-	41.15	28.09	23.51	50.20	56.51	13.30		
	Centralized	-	-	-	80.12	36.41	26.64	96.99	86.83	20.69		
	Local-only	-	-	-	86.25	37.57	27.13	108.66	91.92	21.43		
50%	FedPrLLM	Layer	One-shot	✗	80.31	36.57	26.69	102.71	88.26	20.56		
		Row	One-shot	✗	80.71	36.61	26.64	101.85	88.31	20.55		
		Column	One-shot	✗	4463.92	22138.56	713.56	14256.86	7392.64	407313.84		
		Layer	Iterative	✗	81.22	36.54	26.68	102.72	88.38	20.55		
		Row	Iterative	✗	81.26	36.55	26.64	103.66	88.94	20.60		
		Column	Iterative	✗	4061.96	17610.52	1158.75	13401.63	6941.72	168643.04		
		Layer	One-shot	✓	87.97	37.70	27.27	112.52	92.90	22.21		
		Row	One-shot	✓	88.35	37.72	27.25	112.17	93.07	22.21		
		Column	One-shot	✓	4557.48	29140.28	982.59	12021.08	7801.23	264723.12		
		Layer	Iterative	✓	87.28	37.69	27.27	112.95	92.58	22.39		
		Row	Iterative	✓	87.61	37.60	27.27	113.32	92.61	22.41		
		Column	Iterative	✓	6929.83	15189.83	1178.40	10208.03	5220.64	39172.53		
			Centralized	-	-	-	193.10	71.66	34.94	363.71	220.81	52.42
			Local-only	-	-	-	208.48	82.24	37.27	409.47	271.49	55.39
		60%	FedPrLLM	Layer	One-shot	✗	187.00	74.66	35.38	339.79	241.14	52.61
				Row	One-shot	✗	186.10	74.64	35.47	337.69	242.96	52.61
Column	One-shot			✗	5604.92	31222.37	1338.25	28046.95	7553.32	322022.84		
Layer	Iterative			✗	191.22	72.90	35.83	368.87	237.45	53.78		
Row	Iterative			✗	190.60	73.74	35.77	367.56	235.51	53.25		
Column	Iterative			✗	6785.79	13234.02	1903.66	24022.75	8125.57	46139.19		
Layer	One-shot			✓	216.09	91.63	38.22	429.58	293.11	60.49		
Row	One-shot			✓	215.50	91.60	38.25	428.87	294.44	60.48		
Column	One-shot			✓	7600.58	41079.65	1910.36	18249.40	7601.34	416094.71		
Layer	Iterative			✓	220.22	90.60	38.79	427.12	283.34	61.25		
Row	Iterative			✓	220.16	90.58	38.74	428.36	282.20	61.55		
Column	Iterative			✓	4242.84	11345.68	2133.62	29512.89	7113.24	133467.18		
	Centralized	-	-	-	698.79	299.42	110.70	1902.56	735.73	131.13		
	Local-only	-	-	-	782.42	412.24	144.90	1780.26	863.50	152.97		
70%	FedPrLLM	Layer	One-shot	✗	737.07	366.28	120.33	1521.25	793.55	156.63		
		Row	One-shot	✗	718.37	369.65	118.24	1557.08	792.08	154.72		
		Column	One-shot	✗	18649.81	18136.88	3180.23	49646.82	12010.97	466632.84		
		Layer	Iterative	✗	721.31	355.21	113.31	1675.79	775.69	146.27		
		Row	Iterative	✗	734.43	349.63	113.65	1757.10	767.13	133.92		
		Column	Iterative	✗	28179.23	17249.42	3967.48	29254.5	10233.18	314505.62		
		Layer	One-shot	✓	839.42	484.11	188.18	1633.85	890.27	174.11		
		Row	One-shot	✓	830.33	483.58	187.11	1641.92	891.27	172.74		
		Column	One-shot	✓	26556.95	21627.29	3383.87	54429.17	14951.70	239612.84		
		Layer	Iterative	✓	887.36	469.70	173.86	1789.42	858.48	162.24		
		Row	Iterative	✓	896.85	454.31	172.48	1740.04	879.50	168.51		
		Column	Iterative	✓	8660.95	18472.69	3246.05	11427895.00	8037.55	738685.56		

B.2 MORE RESULTS ON THE COMPARISON GROUP FOR LOCAL PRUNING

The results of changing the comparison group for the local pruning method (i.e., Wanda) are shown in Table 7.

Table 7: Perplexity of pruned LLMs under 50% sparsity ratio when changing the comparison group for the local pruning method (i.e., Wanda). FedPrLLM adopts one-shot pruning and no weight scaling.

Local Compar.		Compar.		LLaMA			LLaMA-2		LLaMA-3	
Group	Dataset	Method	Group	7B	13B	30B	7B	13B	8B	
Layer	WikiText2	Centralized	-	7.94	6.57	5.47	7.38	5.92	12.04	
		Local-only	-	8.16	6.74	5.58	7.56	6.06	12.43	
		FedPrLLM	Layer		7.98	6.60	5.48	7.38	5.95	12.09
			Row		31.85	10.08	11.33	39.07	124.08	17.51
			Column		1749.59	10183.32	541.62	25258.16	5503.91	336255.96
		C4	Centralized	-	10.28	8.63	7.59	10.24	8.49	19.18
	Local-only		-	10.56	8.90	7.86	10.52	8.76	19.64	
	FedPrLLM		Layer		10.34	8.71	7.72	10.32	8.63	19.09
			Row		34.90	12.35	12.75	29.79	207.57	28.05
			Column		975.75	12605.58	553.85	13950.23	4899.58	129415.62
	PTB		Centralized	-	92.84	43.47	27.25	306.71	119.17	23.14
		Local-only	-	99.13	45.34	27.87	338.70	136.88	23.69	
FedPrLLM		Layer		91.99	43.59	27.25	305.79	124.27	22.85	
		Row		284.19	109.14	110.46	1886.94	480.24	44.71	
		Column		3976.21	28144.48	711.16	14131.82	7134.88	293147.84	
Column		WikiText2	Centralized	-	8.86	7.68	5.67	10.41	6.38	83.67
	Local-only		-	8.86	7.68	5.67	10.41	6.38	83.67	
	FedPrLLM		Layer		8.86	7.68	5.67	10.41	6.38	83.67
			Row		138.54	100.80	49.17	764.32	2580.88	400.95
			Column		8.86	7.68	5.67	10.41	6.38	83.67
	C4		Centralized	-	14.10	11.20	8.06	17.90	9.57	30.88
		Local-only	-	14.10	11.20	8.06	17.90	9.57	30.88	
		FedPrLLM	Layer		14.10	11.20	8.06	17.90	9.57	30.88
			Row		155.15	87.03	48.19	222.47	5135.37	327.77
			Column		14.10	11.20	8.06	17.90	9.57	30.88
		PTB	Centralized	-	108.37	47.17	29.22	4567.49	115.68	240.14
	Local-only		-	108.37	47.17	29.22	4567.49	115.68	240.14	
FedPrLLM	Layer			108.37	47.17	29.22	4567.49	115.68	240.14	
	Row			1060.91	394.57	239.91	21323.02	1075.71	928.73	
	Column			108.37	47.17	29.22	4567.49	115.68	240.14	

B.3 MORE RESULTS ON USING SPARSEGPT AS THE LOCAL PRUNING METHOD

For SparseGPT, we utilize the pruning metric proposed in SparseGPT (Frantar & Alistarh, 2023) and do not perform the weight update procedure (also adopted in Wanda (Sun et al., 2024); see Table 7 in (Sun et al., 2024)). The experimental results of using SparseGPT as the local pruning method are shown in Tables 8 and 9. These results show a trend similar to those obtained using Wanda as the local pruning method and further demonstrate the generality of our findings.

Table 8: Perplexity (WikiText2 / C4 / PTB) of pruned LLMs under 50% sparsity ratio using SparseGPT as the local pruning method.

Method	Compar. Group	Prune Stra.	Weight Scaling	LLaMA-7B
Centralized	-	-	-	7.40 / 9.54 / 76.18
Local-only	-	-	-	8.11 / 10.44 / 95.12
FedPrLLM	Layer	One-shot	✗	8.04 / 10.37 / 93.52
	Row	One-shot	✗	8.05 / 10.37 / 93.15
	Column	One-shot	✗	4279.74 / 4868.07 / 11451.43
	Layer	Iterative	✗	8.04 / 10.37 / 94.09
	Row	Iterative	✗	8.06 / 10.37 / 93.15
	Column	Iterative	✗	2562.72 / 4263.29 / 5643.11
	Layer	One-shot	✓	8.17 / 10.52 / 97.55
	Row	One-shot	✓	8.18 / 10.53 / 97.32
	Column	One-shot	✓	6524.84 / 7887.48 / 9790.79
	Layer	Iterative	✓	8.16 / 10.51 / 97.72
Row	Iterative	✓	8.17 / 10.52 / 97.14	
Column	Iterative	✓	2741.71 / 3998.72 / 6088.04	

Table 9: Perplexity (WikiText2 / C4 / PTB) of pruned LLMs under 50% sparsity ratio when changing the comparison group for the local pruning method (i.e., SparseGPT). FedPrLLM adopts one-shot pruning and no weight scaling.

Local Compar. Group	Method	Compar. Group	LLaMA-7B
Layer	Centralized	-	7.91 / 10.21 / 83.25
	Local-only	-	8.89 / 11.58 / 108.47
	FedPrLLM	Layer	8.83 / 11.50 / 106.83
Column	FedPrLLM	Row	183.63 / 134.18 / 913.14
	FedPrLLM	Column	4623.88 / 4722.64 / 12115.99
	Centralized	-	8.86 / 14.10 / 108.37
Column	Local-only	-	8.86 / 14.10 / 108.37
	FedPrLLM	Layer	8.86 / 14.10 / 108.37
	FedPrLLM	Row	138.54 / 155.15 / 1060.91
FedPrLLM	Column	8.86 / 14.10 / 108.37	

B.4 MORE RESULTS ON ZERO-SHOT TASKS

The experimental results on seven zero-shot tasks are shown in Tables 10 and 11.

Table 10: Accuracies (%) on seven zero-shot tasks of pruned LLaMA-7B model under 50% sparsity ratio using Wanda as the local pruning method.

Method	Compar. Group	Prune Stra.	Weight Scaling	HellaSwag	WinoGrande	OBQA	RTE	BoolQ	ARC-c	ARC-e	Mean
Dense	-	-	-	56.96	70.09	34.20	66.43	75.11	41.89	75.29	59.99
Centralized	-	-	-	51.89	66.54	28.60	55.60	71.16	36.86	69.44	54.30
Local-only	-	-	-	51.52	66.23	28.55	55.37	70.85	36.49	69.13	54.02
FedPrLLM	Layer	One-shot	✗	51.93	66.61	29.80	53.49	71.22	37.03	69.49	54.22
	Row	One-shot	✗	51.84	66.61	30.20	53.07	71.16	36.77	69.61	54.18
	Column	One-shot	✗	26.24	50.51	13.60	52.35	38.01	20.65	30.56	33.13
	Layer	Iterative	✗	51.93	66.46	29.20	54.15	71.13	36.95	69.61	54.20
	Row	Iterative	✗	51.90	66.54	29.40	54.33	71.13	36.69	69.44	54.20
	Column	Iterative	✗	26.28	49.96	11.60	52.35	40.55	21.25	31.44	33.35
	Layer	One-shot	✓	51.42	66.51	30.20	53.07	71.19	36.60	68.98	54.00
	Row	One-shot	✓	51.80	66.33	30.20	53.79	71.10	36.30	69.16	54.09
	Column	One-shot	✓	25.92	50.12	12.00	51.26	38.62	20.14	29.46	32.50
	Layer	Iterative	✓	51.90	66.14	28.80	53.79	71.07	36.77	69.40	53.98
	Row	Iterative	✓	51.89	66.54	29.60	54.11	71.15	36.20	69.10	54.08
	Column	Iterative	✓	26.20	49.57	11.80	53.43	38.81	21.42	31.86	33.30

Table 11: Accuracies (%) on seven zero-shot tasks of pruned LLaMA-7B model under 50% sparsity ratio when changing the comparison group for the local pruning method (i.e., Wanda). FedPrLLM adopts one-shot pruning and no weight scaling.

Local Compar. Group	Method	Compar. Group	HellaSwag	WinoGrande	OBQA	RTE	BoolQ	ARC-c	ARC-e	Mean	
Layer	Centralized	-	50.00	66.85	28.40	50.18	69.69	36.60	67.13	52.69	
	Local-only	-	49.59	66.33	27.57	50.07	68.58	35.56	67.11	52.12	
	FedPrLLM	Layer		50.04	65.59	27.80	49.82	68.72	36.09	67.93	52.28
		Row		44.34	64.01	26.40	51.62	56.51	30.55	66.08	48.50
		Column		25.78	50.91	12.20	52.35	37.95	20.82	27.86	32.55
Column	Centralized	-	48.92	65.82	26.20	56.68	65.11	34.56	66.79	52.01	
	Local-only	-	48.92	65.82	26.20	56.68	65.11	34.56	66.79	52.01	
	FedPrLLM	Layer		48.92	65.82	26.20	56.68	65.11	34.56	66.79	52.01
		Row		35.60	56.20	20.80	53.43	50.95	26.28	60.23	43.35
		Column		48.92	65.82	26.20	56.68	65.11	34.56	66.79	52.01

C THE USE OF LARGE LANGUAGE MODELS (LLMs)

We used Large Language Models (LLMs) to enhance the language and clarity of this manuscript. Their role included rephrasing for readability, correcting grammatical errors, and ensuring consistent terminology. All core scientific contributions, including the proposed methods, experimental design, and results analysis, are original to the authors. The LLMs acted solely as writing assistants and did not influence the research ideas or outcomes presented.