electronics

MDPI

Article

Communication-Efficient and Private Federated Learning with Adaptive Sparsity-Based Pruning on Edge Computing

Shijin Song [1], Sen Du [1], Yuefeng Song [1] and Yongxin Zhu [2,*]

[1] School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China; jennifer-song@sjtu.edu.cn (S.S.); sen_du@sjtu.edu.cn (S.D.); songyuefeng@sjtu.edu.cn (Y.S.)
[2] Shanghai Advanced Research Institute, Chinese Academy of Sciences, Shanghai 201203, China
* Correspondence: zhuyongxin@sari.ac.cn

Abstract: As data-driven deep learning (DL) has been applied in various scenarios, the privacy threats have become a widely recognized problem. To boost privacy protection in federated learning (FL), some methods adopt a one-shot differential privacy (DP) approach to obfuscate model updates, yet they do not take into account the dynamic balance between efficiency and privacy protection. To this end, we propose ASPFL—an efficient FL approach with adaptive sparsity-based pruning and differential privacy protection. We further propose the adaptive pruning mechanism by utilizing the Jensen-Shannon divergence as the metric to generate sparse matrices, which are then employed in the model updates. In addition, we introduce adaptive Gaussian noise by assessing the variation of sensitivity through post-pruning uploading. Extensive experiments validate that our proposed ASPFL boosts convergence speed by more than two times under non-IID data. Compared with existing DP-FL methods, ASPFL can maximally achieve over 82% accuracy on CIFAR-10, while the communication cost is greatly reduced by 40% under the same level of privacy protection.

Keywords: federated learning; differential privacy; sparsity; channel pruning

Citation: Song, S.; Du, S.; Song, Y.; Zhu, Y. Communication-Efficient and Private Federated Learning with Adaptive Sparsity-Based Pruning on Edge Computing. *Electronics* **2024**, *13*, 3435. https://doi.org/10.3390/electronics13173435

Academic Editor: Christos J. Bouras

Received: 27 July 2024
Revised: 26 August 2024
Accepted: 28 August 2024
Published: 29 August 2024

1. Introduction

With the vigorous development of deep neural networks (DNNs) for various tasks such as computer vision (CV) [1,2] and natural language processing (NLP) [3], deep learning (DL) has evolved into a technology dependent on large amounts of training data. Recently, the demand for efficient and privacy protection DL solutions has received high attention. Federated learning (FL) has emerged as a promising paradigm of distributed machine learning [4,5]. In contrast to traditional deep learning, which uploads data to the cloud for training, federated learning allows multiple edge devices to locally train the model and upload the model updates.

Nonetheless, model updates dependent on local private samples and large-scale parameters among clients raise significant communication burdens and privacy concerns. As for communication overhead, recent studies have revealed that most deep neural networks (DNN) are over-parameterized. Denil et al. [6] proposed that using only a few weights is enough to achieve similar performance to the MLP and CNN. Michel et al. [7] demonstrated that a significant proportion of the heads in a Transformer can be pruned without substantially degrading the model's performance. A feasible scheme is to reduce parameter transmission by model compression (pruning and distillation) [8], resulting in an appreciably smaller model and faster inference.

Research on the privacy and security issues of FL has revealed that sensitive personal parameters uploaded by clients may be leaked through model inversion attacks [9,10]. Therefore, combining differential privacy (DP) with FL is considered an effective privacy protection technique [11–14]. The mechanism of DP typically involves adding random noise to intermediate outputs in order to ensure that a change in a particular input element

will not have a significant impact on the output distribution [12]. However, in DP-based FL, an inevitable trade-off exists between model utility and privacy protection level. Enhanced privacy preservation measures typically necessitate the introduction of artificial noise with higher variance, which inevitably leads to a degradation in the model's predictive accuracy and response speed.

Accordingly, we urgently need a general and common framework to strengthen data privacy protection and reduce communication costs while quantitatively answering important questions:

> *Can model pruning and differential privacy be orthogonal? How does pruning affect the fine-tuning of the private noise?*

Several studies have attempted to address similar inquiries from various perspectives [15–19], and numerous concepts of privacy and associated strategies have been proposed to tackle related problems. Mireshghallah et al. [15] proposed a method for evolving large models into compressed ones through a differential privacy iterative magnitude pruning (DPIMP) framework. Its main idea is to sort the weights by magnitude, then eliminate the parameters with small magnitudes based on a pruning ratio. This approach can achieve up to 50% parameter sparsity while maintaining model performance. However, the system is configured under central machine learning and not placed in the context of FL, while the pruning ratio is usually determined empirically. Lin et al. [18] proposed a proprietary model compression framework named RONA, which integrates knowledge distillation and differential privacy techniques to achieve efficient deployment of deep learning models on mobile devices. But we cannot guarantee the existence of a large, powerful, and complex teacher model based on sensitive data for multiple tasks. GFL-ALDPA aimed to achieve gradient compression and user privacy protection by jointly using dimensional reduction and local differential privacy (LDP) [19]. However, it has not fully explained and directly quantified the impact of compression on the injected private noise.

To address the challenges of capacity bottlenecks and privacy protection when deploying deep learning models on edge devices, we propose an enhanced private model compression framework based on adaptive DP and develop a learnable sparsity-based pruning module. Model compression and data privacy protection are achieved jointly in this framework. We introduce the compressed local model parameters space through the function of sparsity matrix to reduce the communication overhead. And following rigorous DP, we propose an adaptive differential privacy to not only provide strong privacy protection but also effectively defend against non-adversarial attacks. It can also ensure a high accuracy of model training.

The contributions of this paper are summarized as follows:

(1) We proposed a communication-efficient federated learning framework that achieves a high-accuracy classification model with a privacy guarantee. This framework jointly considers the dynamic balance between the efficiency and privacy issues under non-IID distributions while achieving good model performance.
(2) To enhance communication efficiency, we proposed an adaptive channel pruning mechanism using a Jensen-Shannon divergence feature p_i to generate the sparsity matrix, reducing the rounds required for equivalent performance by more than 2 times.
(3) To ensure the privacy guarantee, we estimate the sensitivity after pruning and introduce a Gaussian noise to the aggregated model. In addition, we give a lower bound of variance σ^2 for adaptive collaborative training. Our method achieves a higher model accuracy (82.42%) and 40% improvement in communication rounds to the baseline methods.

The rest of the paper is organized as follows: Section 2 introduces the context of differential privacy for federated learning and model pruning. Section 3 describes our proposed adaptive and efficient Sparsity-pruning-based federated learning scheme. Section 4 exper-

imentally illustrates the accuracy and performance of our proposed method on CIFAR-10. Section 5 presents our conclusions.

2. Related Works

Federated learning (FL) is a fundamental training paradigm in machine learning that was first proposed in 2016 [20]. It aims to build a collaborative machine learning model on distributed datasets. Recently, federated learning has faced growing challenges, including communication overhead, data heterogeneity, and privacy threats. This section gives the introduction of two topics, namely differential privacy for federated learning and model pruning.

2.1. Differential Privacy for Federated Learning

Differential privacy (DP) has been widely adopted to mitigate user privacy leakage in federated learning [21–23]. Differential Private Stochastic Gradient Descent (DP-SGD) [22] was proposed to provide a data-level privacy guarantee and can be easily achieved by adding Gaussian noise to the clipped gradients. However, FL not only focuses on the privacy protection of local data but also requires information security between clients. Existing works can be summarized into two classes: centralized DP (CDP) [12,21,24] and local DP (LDP) [25–27]. The central idea of LDP is randomized response (RR). In federated learning, LDP can enable parties to scramble their data and then publish the obfuscated data to an untrusted server.

2.2. Model Pruning

With the extensive deployment of DNNs, model pruning has become an important topic to reduce the computational resources. A popular way is magnitude-based pruning [15,28]. Jiang et al. [28] apply the lottery hypothesis, which posits that pruning a network based on the magnitude of its weights, in principle, yields an optimal substructure of the original network.

In general, pruning methods can be divided into two categories: structured pruning and unstructured pruning. Structured pruning reduces the model complexity by removing structural units within the network such as kernels, filters, or layers in different granularities [29–31]. It typically requires specific network structures, which is inconsistent with the "lottery ticket hypothesis". Moreover, selection of the optimal pruning rate mostly depends on the empirical knowledge, and the original model requires retraining. Zhu et al. [30] proposed an effective model compression and acceleration framework, FedLP-Q, through hierarchical pruning and quantization. The pruning process in the frame takes place at each layer of the model, removing unimportant weights or entire layers to reduce the depth and width of the model, but this approach results in fewer feature maps in the intermediate representation.

In contrast, unstructured pruning mainly focuses on pruning at the level of individual weights. Without changing the original structure of the network, it achieves sparsity by setting unimportant weights to zero. Qian et al. [32] propose a dynamic adjustment strategy to reduce unnecessary transmission costs by gradually increasing the sparsity ratio and replacing aggregation weights with the inverse ratio of sparsity. But the authors did not take into account that the unevenness of the data distribution can affect aggregate weights.

3. Our Methodology: ASPFL Framework

In this section, we present a general framework, ASPFL, introduce the adaptive sparsity-based pruning module and central differential privacy mechanism, and perform an extensive privacy analysis of the model after pruning. The notations used in this paper are summarized in Table 1.

Table 1. Descriptions of Main Notations.

Symbol	Descriptions
N	total number of clients
T	total number of global epochs
t	the index of the t-th global epoch
L	number of local epochs for clients in a round
l	the index of the l-th local epoch
$\boldsymbol{g}$	gradient of the model
$\mathcal{D}_i$	local dataset of the i-th client
B	batch size
K	set of participating clients
$\boldsymbol{\theta}_i$	local model parameters of the i-th client
$\boldsymbol{W}_i$	weight matrices of the i-th client
$\|*\|$	size of set
ε	global privacy budget
σ	the noise level
S_f	the sensitivity of the query function
Δ_i	model update of the i-th client
$\boldsymbol{R}$	the sparse matrix
γ	tailoring factor

3.1. Overview and Problem Statement

Workflow: The workflow of our proposed ASPFL is shown in Figure 1. It can be summarized into five stages: (1) The server first performs global model initialization and distributes it to the clients. (2) Sampled clients train their local models based on the current global model and their respective datasets. During local training, each client executes local model updates and gradient clipping based on L_2-norm to meet the requirements of DP. (3) After L rounds' local training, each client performs channel pruning according to a learnable sparsity matrix and estimates the sensitivity of all local model unions. Then updates of the local model are uploaded back to the server. (4) The server collects all model updates and introduces an adaptive noise based on the estimated sensitivity to perform model aggregation. (5) Finally, a new global model is obtained. Below, we will explain the specific operation of each stage in detail.

System Model: Based on the client/server (C/S) framework, the basic FL system comprises N clients and a server aiming to collaboratively train an optimal model while achieving privacy protection. Assume that the server is honest but curious, client c_i holds the local dataset $\mathcal{D}_i$ with $|\mathcal{D}_i|$ samples, where $i \in \{1, 2, \ldots, N\}$. The union of all local datasets is $\mathcal{D} := \bigcup \mathcal{D}_i$. The objective of our ASPFL is to aggregate local models from N clients and obtain an optimal global model $\boldsymbol{\theta}$ that minimizes the global empirical risk:

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{N} p_i \mathcal{L}(\boldsymbol{\theta}_i), \tag{1}$$

where $\mathcal{L}(\boldsymbol{\theta}_i)$ is a non-negative and convex loss function on each client, $\mathcal{L}(\boldsymbol{\theta})$ is the global loss function. In FedAvg [4], p_i denotes the aggregation weight, which can be calculated by the dataset size $\frac{|\mathcal{D}_i|}{|\mathcal{D}|}$.

Threat model: Assume that the server is honest but curious; it adheres to federation rules while maintaining a curiosity towards the local data of each client. Although FL keeps individual datasets locally, sharing model parameters with the server may expose clients' private information through model-inversion attacks. These attacks include attempts by the server to recover training datasets or infer private features based on uploaded parameters and deduce whether a sample belongs to a client's training dataset from output differences. Furthermore, each client can also be considered honest but curious. They potentially perform similar privacy attacks on other clients' data after receiving broadcast model parameters.

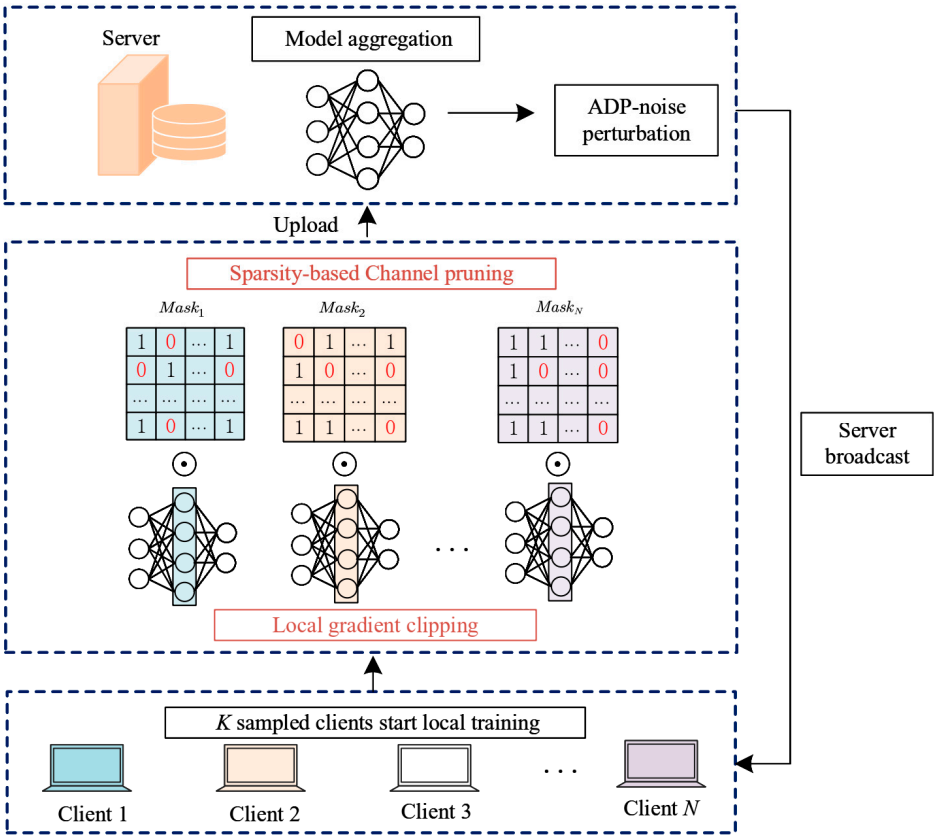


Figure 1. Overview of our proposed ASPFL scheme.

3.2. ASPFL Algorithm

Figure 1 shows the workflow of our proposed scheme, consisting of 5 steps. We show the specific procedures in Algorithm 1, then illustrate each step below.

(1) **Global model initialization.** The server first initializes global model parameters $\boldsymbol{\theta}_0$ (Line 1 in Algorithm 1) and distributes $\boldsymbol{\theta}_0$ to each client.

(2) **Local models update.** In the global round t, K clients are randomly selected to participate in local training. Local model parameters $\boldsymbol{\theta}_{i,0}^t$ is first initialized as the global model parameters $\boldsymbol{\theta}_{i,0}^t \leftarrow \boldsymbol{\theta}^{t-1}$. Then L steps local training are performed to update local model parameters $\boldsymbol{\theta}_{i,l}^t$. We designed the gradient clipping process of incorporating tailoring factors γ into the stochastic gradient descent (SGD) optimizer to obtain a local model $\boldsymbol{\theta}_{i,l}^t$. A tailoring factor $\gamma_{i,l}^t$ is applied to control the magnitude of parameter updates, which is represented as:

$$\gamma_{i,l}^t = \min\left(1, \frac{C}{\left\|\boldsymbol{g}_{i,l}^t\right\|_2}\right), \tag{2}$$

where $\boldsymbol{g}_{i,l}^t$ denotes the gradient at step l of round t, and C is the upper bound of gradients to prevent instability in the training regimen. Specifically, for the client i, the local gradient clipping process is formulated as:

$$\tilde{\boldsymbol{g}}_{i,l}^t = \gamma_{i,l}^t \boldsymbol{g}_{i,l}^t = \min\left(1, \frac{C}{\left\|\boldsymbol{g}_{i,l}^t\right\|_2}\right) \nabla \mathcal{L}\left(\boldsymbol{\theta}_{i,l-1}^t; B_{i,l}^t\right), \tag{3}$$

$$\boldsymbol{\theta}_{i,l}^t = \boldsymbol{\theta}_{i,l-1}^t - \eta \tilde{\boldsymbol{g}}_{i,l}^t, \tag{4}$$

where $B_{i,l}^t$ is a random batch of data samples and η denotes the learning rate.

(3) **Sparsity-based channel pruning.** In the round t, the client i finishes L steps local training and obtains a new local model $\boldsymbol{\theta}_{i,L}^{t}$. To reduce the communication overhead required for uploading parameters, we employ an adaptive sparse matrix to discard the less significant parts of the model updates. A sparse matrix $\boldsymbol{R}_i$ composed of elements "1" and "0" is allocated for the client i. By Hadamard product of model updates with this matrix, the less important weight parameters will be zeroed out. The client i combines the accumulated gradients $\boldsymbol{\theta}_{i,L}^{t} - \boldsymbol{\theta}^{t-1}$ with $\boldsymbol{R}_i$ and the uploaded part is formulated as:

$$\Delta\boldsymbol{\theta}_i = (\boldsymbol{\theta}_{i,L}^{t} - \boldsymbol{\theta}^{t-1}) \bigodot \boldsymbol{R}_i = \eta \sum_{l=1}^{L} \gamma_l^t \boldsymbol{g}_{i,l}^{t} \bigodot \boldsymbol{R}_i. \tag{5}$$

The selection method for the mask matrix $\boldsymbol{R}_i$ will be elaborated in the next subsection. After channel pruning, the sensitivity of uploading model parameters S_f can be estimated as:

$$S_f = 2\eta CLp_i \tag{6}$$

where p_i is a personalized parameter for the client i.

(4) **Model aggregation and noise perturbation.** Initially, the server receives all model updates from participating clients and computes the weighted average of model updates, yielding a preliminary version of the aggregated global model. To fortify privacy preservation via differential privacy, the server introduces adaptive Gaussian noise onto the preliminary global model. The variance σ^2 of this noise is judiciously determined based on the desired privacy budget (ε and δ) and the sensitivity S_f. Mathematically, the noise-injected global model is denoted as:

$$\boldsymbol{\theta}^{t} = \boldsymbol{\theta}^{t-1} + \left(\sum_{i=1}^{N} \frac{|\mathcal{D}_i|}{|\mathcal{D}|} \Delta\boldsymbol{\theta}_i \right) + \mathcal{N}\left(0, \sigma^2 \boldsymbol{I}_d\right). \tag{7}$$

(5) **Global model broadcasting.** Upon completion of the global model update in round t, the server is responsible for broadcasting the newly generated global model $\boldsymbol{\theta}^t$ to all participating clients, thereby initiating the subsequent round of local training processes as $\boldsymbol{\theta}_{i,0}^{t+1} \leftarrow \boldsymbol{\theta}^{t}$.

The procedures of our ASPFL algorithm have been concluded in Algorithm 1. The primary improvements of this algorithm from traditional FL include three aspects: (1) Scaling rule of accumulated gradients during local training. (2) Adaptive channel pruning based on sparse matrices for non-IID data distributions. (3) Post-pruning noise addition strategy at the server side.

3.3. Adaptive Sparsity-Based Channel Pruning

To mitigate client dependence on network stability during parameter uploads, we propose a method of achieving model sparsity through the introduction of learnable gating parameters to regulate the extent of model compression. To attain model sparsity within the context of a federated learning architecture, we incorporate a normalization technique that entails the direct application of a mask matrix, denoted as $\boldsymbol{R}_i$, onto the model parameter. The mask matrix $\boldsymbol{R}_i$ is derived via the gated parameter p_i, which serves as a quantitative indicator of the proportion of "1"s contained within $\boldsymbol{R}_i$. This operation effectively prunes the model by zeroing out coefficients based on the binary values of $\boldsymbol{R}_i$, leading to a decrease in the bandwidth requirements but also diminishing potential vulnerabilities to model inversion or membership inference attacks.

In the layer-wise pruning-quantization scheme for efficient federated learning proposed in the literature [29,30], p_i represents the probability that the gradient of each layer will be preserved, catering to the demands of communication efficiency. Specifically, Zhu et.al. [29,30] proposed that p_i is typically set by the central server following three steps:

- First set a fixed value p;
- Then, specify the layer-preserving rate (LPR) that satisfies the *Bernoulli(p)* distribution;
- Finally, prune the cumulative gradients according to LPR.

Algorithm 1 ASPFL

Initialization: tailoring threshold C, local models, local datasets $\mathcal{D}_i$, privacy configurations (ε, δ), etc.

for global round $t = 1, 2, \ldots, T$ **do**
  for participating client i in parallel **do** ※client side
    for each step $l = 1, 2, \ldots, L$ **do**
      Clip local gradient: $\overset{\sim}{\boldsymbol{g}}_{i,l}^{t}$;
      Update $\boldsymbol{\theta}_{i,\,l}^{t} \leftarrow Local\_Update\left(\boldsymbol{\theta}_{i,\,l-1}^{t}; \mathcal{D}_i\right)$;
    end
    Quantize the JS distribution feature: p_i;
    Generalize the mask matrix: $\boldsymbol{R}_i = \{r_{jk}\}$, $r_{jk} \sim Bernouli(p_i)$;
    Conduct sparsity pruning: $\left(\boldsymbol{\theta}_{i,\,L}^{t} - \boldsymbol{\theta}^{t-1}\right) \odot \boldsymbol{R}_i$;
    Upload model updates to the server;
  end
  Aggregate updates and calculate the sensitivity S_f; ※server side
  Introduce Gaussian noise to $\boldsymbol{\theta}^{t}$;
  Download the global model: $\boldsymbol{\theta}_{i,\,0}^{t+1} \leftarrow \boldsymbol{\theta}^{t}$; ※client side
end

Output: global model: $\boldsymbol{\theta}^{t}$.

In this strategy, the parameter p is designed to be a constant applicable to all clients in model homogeneity cases. Essentially, p is an empirical parameter that may require a lot of prior knowledge and many attempts to determine. However, federated learning faces a critical challenge due to non-independent and identically distributed (non-IID) data [33]. If the nature of non-IID data is not considered, potential biases in capturing the feature representations across various clients will occur, further diminishing the model's accuracy and generalization ability.

Based on this intuition, we integrate the characteristic of data distribution into the sparse matrix selection and propose an adaptive channel pruning strategy. There exists a typical scenario of label skew in federated learning with non-IID data, which may lead to performance degradation of the model on some clients. The disparity in label distribution across clients can be quantified through the probability of each label within their respective datasets. To this end, the distribution of labels from a given client's dataset is extracted and denoted by $P = \{P_i\}$, $i \in N$. Then, a uniform distribution Q is established as a benchmark for comparison. For an aggregated dataset $\mathcal{D}$ encompassing ‡ distinct label categories, a perfectly uniform distribution Q implies that each category is endowed with an equal probability of occurrence, precisely $Q = \left\{\frac{1}{z}\right\}$.

We adopt the Jensen-Shannon (JS) divergence as a metric to quantify the difference between each client's data label distribution P_i and a standard uniform distribution Q. JS divergence is a well-established measure for assessing the similarity, which can be formulated as:

$$D_{KL}(P_i \| Q) = \sum_x P_i(x) \log \frac{P_i(x)}{Q(x)} \tag{8}$$

$$D_{JS}(P_i \| Q) = \frac{1}{2} D_{KL}\left(P_i \middle\| \frac{P_i + Q}{2}\right) + \frac{1}{2} D_{KL}\left(Q \middle\| \frac{P_i + Q}{2}\right) \tag{9}$$

where D_{KL} denotes the Kullback-Leibler (KL) divergence. By calculating the JS divergence between P_i and Q in Equations (8) and (9), we can quantify how far each client's label distribution deviates from a uniform distribution. For each client, we utilize each local dataset's

JS divergence as a distinctive feature p_i, thereby instituting the following procedures to construct adaptive local sparse matrices:

- **Extraction of Label Distributions:** Initially, the label distributions pertinent to each client's local dataset are meticulously extracted and quantified.
- **Computation of JS Divergence:** the JS divergence between the client's label distribution and a predefined uniform distribution is calculated to quantify the similarity. Since the range value of JS divergence is in $[0, ln2]$, we make a linear transformation of the feature p_i following:

$$p_i = \frac{JS_{max} - JS}{JS_{max} - JS_{min}} = \frac{ln2 - JS}{ln2} \tag{10}$$

- **Normalization and Weighting:** The resulting p_i should be normalized to ensure comparability across different clients.
- **Matrix Construction:** The sparse matrix $\boldsymbol{R}_i$ is sampled from the Bernoulli (p_i) distribution for each client. This matrix reflects the unique statistical characteristics of the client's data, particularly focusing on the sparsity patterns that emerge from the label distribution disparities.

This parameter p_i essentially governs the sparsity level of the matrix, precisely quantifying the probability of retaining the element "1" among the matrix elements. In this scenario, mask matrix $\boldsymbol{R}_i$ is a 3D tensor composed of multiple 2D mask matrices, each of which matches the shape of individual slices within $\boldsymbol{W}_i$. The elements $r_{jk}\left(\equiv r_{i,njk}\right)$ of each mask matrix $R_{i,\,n}$ are independently sampled from a *Bernoulli* (p_i). Here, n indexes the tensor, and j and k index the rows and columns of the matrix. Specifically, each element r_{jk} in $R_{i,n}$ can be formulated as:

$$r_{jk} = \begin{cases} 1 \text{ with probability } p_i, \\ 0 \text{ with probability } 1 - p_i. \end{cases} \tag{11}$$

For the n-th weight tensor, we can get $\boldsymbol{W}_{i,n} = \boldsymbol{W}_{i,n} \odot \boldsymbol{R}_{i,n}$ $(w_{jk} = w_{jk} \times . r_{jk})$. If $r_{jk} = 1$, then the corresponding weight w_{jk} in $\boldsymbol{W}_i$ will be retained. Conversely, if $r_{jk} = 0$, w_{jk} will be set to zero or effectively ignored. The generation process of the mask tensor $\boldsymbol{R}_i$ has been illustrated in Figure 2. The process of generating $\boldsymbol{R}_i$ through p_i ensures that the resultant matrix selectively preserves critical information while discarding less relevant data points. By adopting this approach, we can enhance the adaptability and effectiveness of FL algorithms in scenarios characterized by non-IID, thus improving overall model performance and convergence rates.

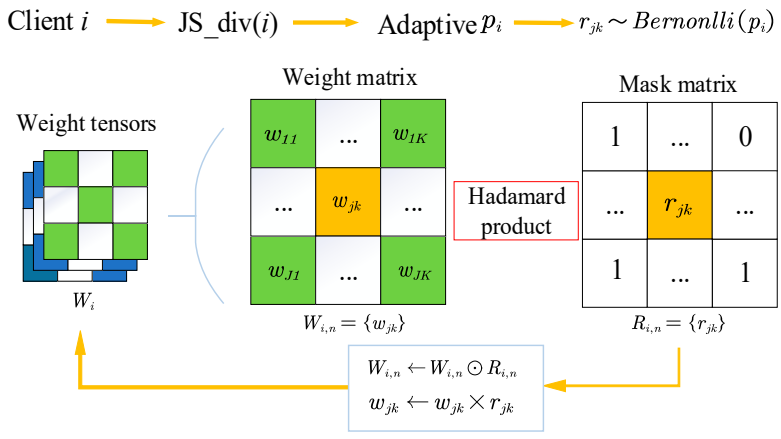


Figure 2. The pipeline of generating the mask $\boldsymbol{R}_i$ and updating weight $\boldsymbol{W}_i$.

3.4. Privacy Guarantee after Pruning

To address the privacy and security concerns of participants, we implement the Gaussian mechanism to achieve DP. This approach is instrumental during the aggregation phase at the central server, where parameter updates from local models are synthesized, thereby effectively mitigating the risk of privacy leakage. Compared to traditional privacy preservation mechanisms, differential privacy offers the advantages of quantifiable privacy budgets and the concealment of sensitive behaviors at the user level. Therefore, we introduce the basic knowledge of DP, which is defined as follows:

Definition 1 [(ε, δ)-DP [23]]**.** *For two adjacent datasets $\mathcal{D}$ and $\mathcal{D}'$ that differ by only one record, a randomized mechanism $\mathcal{M}$ can preserve (ε, δ)-DP if and only if any output $S \subset Range(\mathcal{M})$ holds:*

$$Pr[\mathcal{M}(\mathcal{D}) \in S] \leq Pr\left[\mathcal{M}\left(\mathcal{D}'\right) \in S\right] \times e^{\varepsilon} + \delta \tag{12}$$

where $\varepsilon > 0$ is the privacy budget and $\delta \in (0,\ 1)$ means the failure probability. Equation (12) is also termed the Gaussian mechanism in DP.

During each round of updating information from local models, noise should be introduced, obeying the global privacy budget ε and the sensitivity S_f. The Gaussian mechanism stipulates to add Gaussian noise $\mathcal{N}\left(0,\ \sigma^2\right)$ to each output of the query function, where σ^2 must satisfy the following equation:

$$\sigma^2 \geq \frac{2\ln\left(\frac{1.25}{\delta}\right)}{\varepsilon^2} S_f^2 \tag{13}$$

A. *L_2-norm Sensitivity*

Now we specifically analyze the influence of the channel pruning module on the sensitivity of upload model parameters so as to customize appropriate noise to achieve (ε, δ)-DP. First, we discuss the L_2-norm Sensitivity evaluation of the local parameter updating. The introduction of γ in Equation (2) is to restrict the range of accumulated gradients. If $\left\|\boldsymbol{g}_{i,\,l}^t\right\|_2 > C$, then $\gamma < 1$, and $\widetilde{\boldsymbol{g}} = \gamma \boldsymbol{g}_{i,l}^t$ to prevent gradient explosion. If $\left\|\boldsymbol{g}_{i,l}^t\right\|_2 \leq C$, then $\gamma = 1$ and $\widetilde{\boldsymbol{g}} = \boldsymbol{g}_{i,l}^t$ remains unchanged. Thus, the L_2-norm of $\widetilde{\boldsymbol{g}}$ can be formulated as:

$$\left\|\widetilde{\boldsymbol{g}}\right\|_2 = \begin{cases} C,\ C < \left\|\boldsymbol{g}_{i,l}^t\right\|_2 \\ \left\|\boldsymbol{g}_{i,l}^t\right\|_2,\ C \geq \left\|\boldsymbol{g}_{i,l}^t\right\|_2 \end{cases} \tag{14}$$

It is observed that the clipped gradient $\widetilde{\boldsymbol{g}}$ for each client is bounded by C, i.e., $\left\|\widetilde{\boldsymbol{g}}\right\|_2 \leq C$, for any i, t, and l.

On this basis, the client i completes channel pruning with their respective $\boldsymbol{R}_i$ after L steps local training. The accumulated gradients after pruning are then uploaded to the server. In the whole process, the sensitivity of the gradient uploading function is calculated as follows:

Lemma 1. *For the round t at client i, the L_2-norm sensitivity of uploading after pruning function is*

$$S_f = \max_{\mathcal{D},\mathcal{D}'} \left\|M_i \boldsymbol{G}_i^t\left(\mathcal{D}\right) - M_i \boldsymbol{G}_i^t\left(\mathcal{D}'\right)\right\|_2 = 2\eta C L p_i \tag{15}$$

where M represents the Hadamard product of sparse matrix $\boldsymbol{R}_i$ and $\boldsymbol{G}$ is the accumulated gradients.

Proof. In the round t, client i performs L steps of local clipping SGD to achieve $\boldsymbol{\theta}_i^t$ following:

$$\boldsymbol{\theta}_i^t = \boldsymbol{\theta}^{t-1} - \textstyle\sum_{l=1}^{L} \eta \gamma_{i,\,l}^t \boldsymbol{g}_{i,l}^t \tag{16}$$

Since r_{jk} is independently sampled from *Bernoulli* (p_i), then $E[\boldsymbol{R}_i]$ is p_i. Consequently, under certain conditions (e.g., large sample size or independent sampling), the actual sensitivity can be approximated by its expectation $S_f \approx E\left[S_f\right]$. The expectation of S_f can be represented as:

$$\begin{aligned} E[S_f] &= \max_{\mathcal{D},\mathcal{D}'} E[\|M_i \boldsymbol{G}_i^t(\mathcal{D}) - M_i \boldsymbol{G}_i^t(\mathcal{D}')\|_2] \\ &= \max_{\mathcal{D},\mathcal{D}'} \left\| E[\boldsymbol{R}_i]\boldsymbol{G}_i^t(\mathcal{D}) - E\left[\boldsymbol{R}_i\right]\boldsymbol{G}_i^t(\mathcal{D}') \right\|_2 \\ &= p_i \max_{\mathcal{D},\mathcal{D}'} \left\| \boldsymbol{G}_i^t(\mathcal{D}) - \boldsymbol{G}_i^t(\mathcal{D}') \right\|_2 \end{aligned} \tag{17}$$

Since $\gamma_{i,l}^t \in (0,1]$,

$$\begin{aligned} \left\| \boldsymbol{G}_i^t(\mathcal{D}) - \boldsymbol{G}_i^t(\mathcal{D}') \right\|_2 &= \left\| \sum_{l=1}^{L} \eta \gamma_{i,\,l}^t \boldsymbol{g}_{i,l}^t(\mathcal{D}) - \sum_{l=1}^{L} \eta \gamma_{i,\,l}^t \boldsymbol{g}_{i,l}^t(\mathcal{D}') \right\|_2 \\ &\leq \eta \left\| \sum_{l=1}^{L} \tilde{\boldsymbol{g}}_{i,l}^t(\mathcal{D}) - \sum_{l=1}^{L} \tilde{\boldsymbol{g}}_{i,l}^t(\mathcal{D}') \right\|_2 \end{aligned} \tag{18}$$

Then, the upper bound of the sensitivity is:

$$S_f \approx E\left[S_f\right] \leq \eta p_i \sum_{l=1}^{L} \left\| \tilde{\boldsymbol{g}}_{i,l}^t(\mathcal{D}) - \tilde{\boldsymbol{g}}_{i,l}^t(\mathcal{D}') \right\|_2 \leq 2\eta C L p_i. \tag{19}$$

Given that $\mathcal{D}$ and $\mathcal{D}'$ differ by only one sample, the difference in gradient estimates, $\left\| \tilde{\boldsymbol{g}}_{i,l}^t(\mathcal{D}) - \tilde{\boldsymbol{g}}_{i,l}^t(\mathcal{D}') \right\|_2$, reflects the sensitivity of the gradients to individual samples. Specifically, this difference captures the sum of gradients produced by performing L local iterations on that single sample. Given the constant C, which characterizes the maximum change in the gradient estimates, the upper bound on the sensitivity can be conservatively estimated as $2\eta C L p_i$. □

B. *Privacy Guarantee in Each Round*

Privacy Amplification Theorem [34] states that if a mechanism $\mathcal{M}$ is (ε, δ)-DP over a dataset $\mathcal{D}$ with size m and a subsampling mechanism is used to draw a subset $\mathcal{D}_s$ with size n ($n \leq m$) uniformly at random from $\mathcal{D}$, then applying $\mathcal{M}$ to the subset $\mathcal{D}_s$ guarantees $(\varepsilon', (n/m)\delta)$-DP, where ε' is defined as $\varepsilon' = \log(1 + (n/m)(e^{\varepsilon} - 1))$.

Next, we establish the lower bound of the noise variance σ^2 for each client to achieve (ε, δ)-DP in each round. This provides a valuable suggestion for introducing the appropriate noise for DP-FL. Given that the local updates of clients are conducted on random batches, we can apply the Privacy Amplification Theorem [34] to mitigate the noise variance and achieve (ε, δ)-DP.

Lemma 2. *For each round t at client i, let $q_i = (Lb/|\mathcal{D}_i|)$ ($q_i \leq 1$), and each batch is sampled without replacement, the Gaussian noise with variance σ^2 satisfying the condition*

$$\sigma^2 \geq 32\left(\frac{\eta C L q_i p_i}{\varepsilon}\right)^2 \ln\left(\frac{1.25}{\delta}\right) \tag{20}$$

can achieve $(\varepsilon,\ \delta)$-DP.

Proof. In the round t, client i performs L steps of local clipping SGD, where each batch b is sampled without replacement from its local dataset $\mathcal{D}_i$. Let $q_i = (Lb/|\mathcal{D}_i|)$ ($q_i \leq 1$), the Gaussian noise with variance $\sigma^2 = \frac{2\ln\left(\frac{1.25}{\delta}\right)}{\varepsilon^2} S_f^2$ can achieve $(\log(1 + q_i(e^{\varepsilon} - 1),\ q_i\delta))$-DP

based on Equation (13) and Privacy Amplification Theorem [34]. Since $\varepsilon \in (0, 1)$, we can apply the following inequality:

$$\ln\left(1 + q_i(e^{\varepsilon} - 1)\right) \leq q_i(e^{\varepsilon} - 1) \leq 2q_i\varepsilon \tag{21}$$

Thus, we can deduce that, given ε and δ, the Gaussian noise with variance $\sigma^2 = \frac{2\ln\left(\frac{1.25}{\delta}\right)}{\varepsilon^2} S_f^2$ guarantees at least $(2q_i\varepsilon,\ q_i\delta)$-DP. To achieve (ε, δ)-DP in each global round t, the low bound for σ^2 can be reduced to

$$\sigma^2 = \frac{8{q_i}^2 \ln(1.25/\delta)}{\varepsilon^2} S_f^2. \tag{22}$$

Since $S_f \leq 2\eta CLp_i$ in Equation (19), we can infer that the low bound for σ^2 is

$$\sigma^2 = 32\left(\frac{\eta CLq_i p_i}{\varepsilon}\right)^2 \ln\left(\frac{1.25}{\delta}\right) \tag{23}$$

If $p = \max_i p_i$, we can derive that the lower bound $\sigma_{min} = 4q_i \frac{\eta CLp}{\varepsilon} \sqrt{2\ln\left(\frac{1.25}{\delta}\right)}$. □

3.5. Computational Complexity Analysis

The primary computational cost arises from the local model updates and the sparsity-based pruning processes. Accordingly, this section focuses primarily on analyzing the computational cost introduced by these two critical stages.

In the stage of local model training, the tailoring factor $\gamma_{i,l}^t$ is computed based on the $\left\|\boldsymbol{g}_{i,l}^t\right\|_2$ which involves summing the squares of all elements and taking the square root. Its complexity is $O(d_j)$, where d_j is the dimensionality of the gradient vector for layer j. The process of multiplying each gradient vector $\boldsymbol{g}_{i,l}^t$ by a scalar has a one-time computational cost, which can be negligible. Considering that there are L iterations and J layers, the overall computational complexity of computing the scaling factors and updating the gradients is $O(L*\sum_{j=1}^{J} d_j)$.

In the stage of sparsity-based channel pruning. To compute the JS divergence for client i, we first need to determine the probability of each class. This step has a minimum complexity of is $O(|\mathcal{D}_i|)$ since we need to iterate over each sample and update the counts for each class. Optimization techniques such as using a hash table or dictionary and parallel processing can improve the efficiency of the process. Then, a vector P is constructed to store the probability of each class with a dimension equal to the number of class n. According to Equation (8), we need to access each element in vectors P and Q to compute KL divergence, so the complexity is $O(n)$. Since addition operations have $O(1)$ complexity, the overall complexity of computing JS divergence in Equation (9) is $O(n + 1)$, approximately $O(n)$. Next, we perform linear transformation and normalization to get p_i per client, which has $O(1)$ complexity as it involves a fixed number of simple arithmetic operations. Consequently, we sample a matrix $\boldsymbol{R}_i$ from the $Bernoulli(p_i)$ distribution. Acquiring a binary mask based on p_i introduces a relatively low computation cost, as Bernoulli sampling can be performed efficiently. The element-wise multiplication of the mask matrix $\boldsymbol{R}_i$ and the accumulated gradients is an intrinsic operation and introduces a negligible computational overhead.

Incorporating Gaussian noise introduces additional computations during the global model aggregation process, which is executed on the server side. The complexity of computing the lower bound σ_{min} is $O(n)$, according to Equation (20), as it needs to find the maximum value within the vector p_i. The generation of Gaussian noise and the noise addition typically have a constant time complexity of $O(1)$ depending on the random number generator used. These operations do not significantly impact the total computational overhead.

In summary, the main computational complexity comes from the computation of gradient norms and the counting of sample labels. In Python 3.8.18, the use of specialized libraries such as NumPy 1.24.3, Pandas 2.0.3, and PyTorch 2.1.2 can significantly accelerate the counting of sample labels and the computation of norms, reducing the overall computational complexity of the algorithm.

4. Results and Discussion

In this section, some experiments on the CIFAR-10 and Fashion-MNIST datasets are conducted to evaluate the performance of the proposed ASPFL algorithm.

4.1. Experimental Settings

The experimental results mentioned below are obtained on a computer with an AMD Ryzen 5 5600 6-core processor and 16 GB of RAM. Additionally, the computer is equipped with an NVIDIA GeForce RTX 3070 with 16 GB of memory.

The ratio of the training set to the test set is 8:2. The performance of the model is measured under non-IID distributions. Specifically, we utilize the Dirichlet (α) to simulate the dataset distributions across different clients. We adjust the degree of non-IID through the parameter α ($\alpha > 0$). A larger α indicates that the data sets are closer to being IID. To deploy the model on edge devices, we chose ResNet18 [35] as the backbone model, which mainly functions by multiple convolution layers. In the FL system, the number of clients N = 10 with various participating ratios. The initial learning rate η is set to 0.01 and the epochs of local training L is fixed at 3. We select the threshold C for local gradient clipping as 1. As for the differential privacy, the privacy budget ε is pre-determined to 2.0 and $\delta = 10^{-5}$. We specify the maximum number of global iterations T as 200.

4.2. Pruning Performance Comparison

We first evaluated the performance of the pruning module in ASPFL without introducing the DP noise, hereafter denoted as ASPFL-p. With Dirichlet non-IID data, α is set to 0.5, 0.7, 1, and 10. As depicted in Figure 3, when $\alpha = 0.5$, the distribution of sub-datasets across clients has shown the extreme imbalance numbers and severe label skew for non-IID data. It can be observed that some clients may have significantly high numbers of samples for a particular category and no samples for some categories, resulting in a skewed distribution.

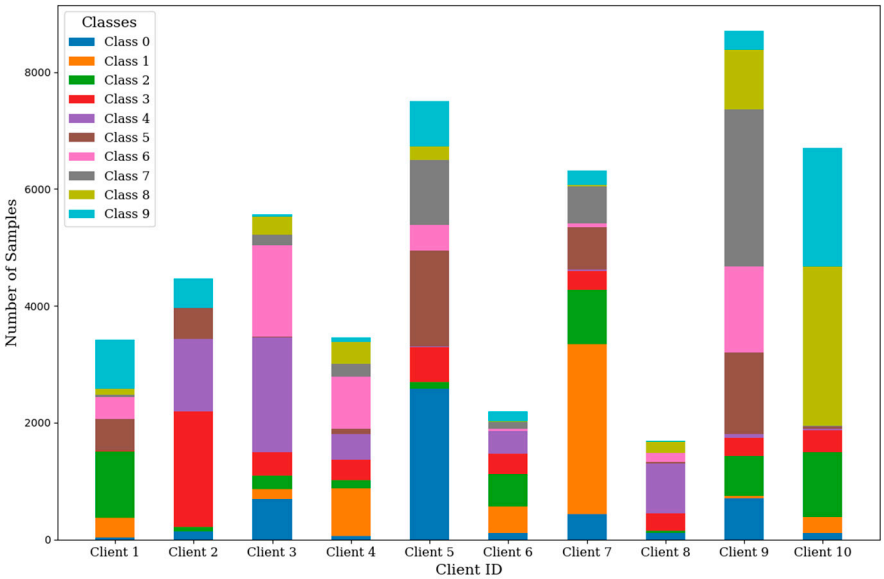


Figure 3. Samples distribution on CIFAR-10 across clients with Dirichlet ($\alpha = 0.5$).

Figures 4 and 5 show that the ASPFL-p scheme achieved comparable accuracy and convergence speed to the original FedAvg [4] and FedLP [29] schemes. For instance, in

Figures 4a and 5a, when the data distribution is highly imbalanced ($\alpha = 0.5$), ASPFL-p exhibits a smoother accuracy curve and faster convergence compared to the other two methods, without significant fluctuations. This suggests that our algorithm maintains stable performance during training, even when faced with extremely imbalanced data distributions. This phenomenon may be attributed to the fact that our algorithm incorporates L_2-norm clipping of accumulated gradients, which helps reduce the volatility of model updates and improve convergence stability on non-IID datasets.

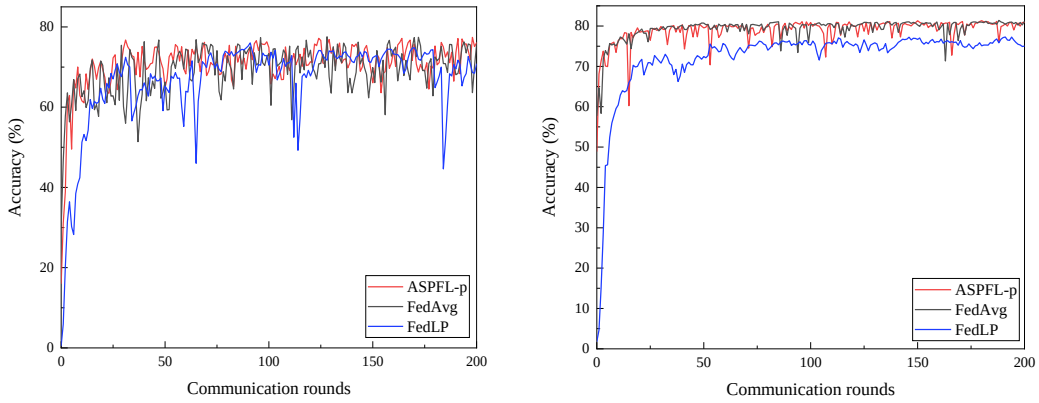


Figure 4. Test accuracy on CIFAR-10 under different Dirichlet (α). ASPFL-p vs. FedAvg [4] vs. FedLP [29]. (**a**) $\alpha = 0.5$; (**b**) $\alpha = 10$.

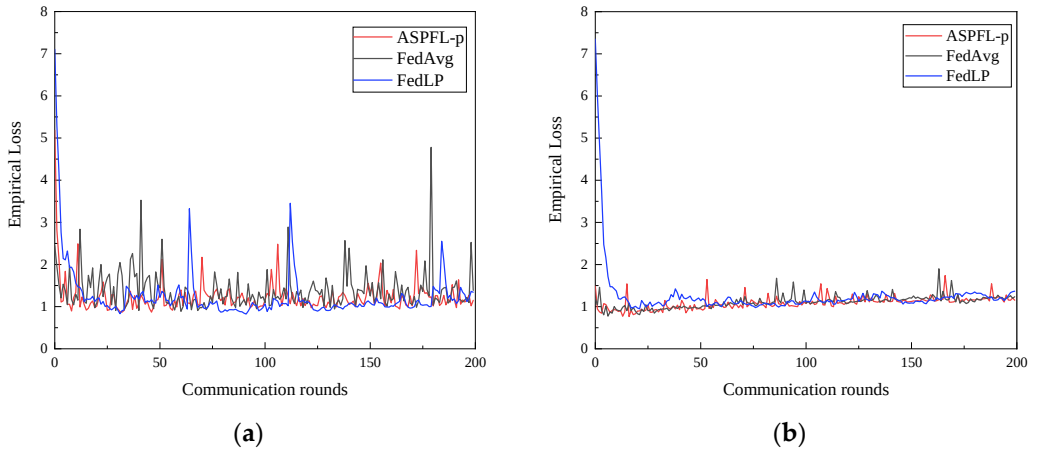


Figure 5. Empirical loss on CIFAR-10 under different Dirichlet (α). ASPFL-p vs. FedAvg [4] vs. FedLP [29]. (**a**) $\alpha = 0.5$; (**b**) $\alpha = 10$.

Table 2 illustrates the comparison results on accuracy and communication overhead. Under different non-IID data distributions characterized by α, ASPFL-p outperforms both FedAvg [4] and FedLP [29] in terms of accuracy. This is because our approach introduces an adaptive strategy that extracts the JS divergence as a feature to ensure the utility of the model. Additionally, the value of α also affects model performance; a larger α leads to a data distribution closer to IID, with a more balanced distribution of samples across clients, thus improving the performance of the aggregated global model. As for communication overhead, ASPFL-p also achieves lower communication rounds and smaller transmission parameters compared to the other two methods. In standard FL settings, models are typically fully connected without specific pruning techniques aimed at reducing model size. ASPFL-p introduces an adaptive pruning mechanism that is related to the data distribution feature to decrease the number of channels within the model. This approach considers the non-IID nature of the data, which may lead to reduced efficacy in model learning

and introduce potential biases into the training process. Certain features might be more pronounced or significant in specific scenarios, such as medical data, thereby necessitating an adaptive approach to matrix design that accounts for these distributional disparities.

Table 2. Comparisons on accuracy, communication rounds, and parameter saving.

Schemes	α	Max acc %	Rounds for 80% acc
FedAvg [4]	0.5	74.32	7
	0.7	77.32	5
	10	80.34	4
FedLP [29]	0.5	74.33	15
	0.7	75.4	14
	10	76.79	12
ASPFL-p (ours)	0.5	76.75	5
	0.7	78.01	4
	1.0	79.22	3
	10	81.32	2

4.3. Privacy Performance Comparison

We assess the privacy performance of ASPFL with IID and Dirichlet (α = 15) non-IID data. We compare the proposed ASPFL with other DP-FL methods, which are either typical CDP-based [12,13] or LDP-based [25] in federated learning. Experimental results show that our ASPFL method has better performance compared with other methods. We use flat clipping with a constant clipping threshold C = 1.

Next, we report the evaluation results on two image datasets: CIFAR-10 and Fashion-MNIST and provide a detailed explanation of the ASPFL framework. For the CIFAR-10 dataset, our proposed scheme demonstrates competitive accuracy across IID and non-IID. We set the ratio of participated clients to be 20%. The average test accuracy over all clients is reported in Table 3. The accuracy under the IID setting (83.57%) is notably higher than DP-FedAvg [24] (82.68%), f-DP [25] (80.37%), and Wu et al. [13] (81.25%). Under non-IID settings, our method achieves higher accuracy (82.43%) than DP-FedAvg [24] (81.12%), f-DP [25] (78.84%), and Wu et al. [13] (79.05%) as well.

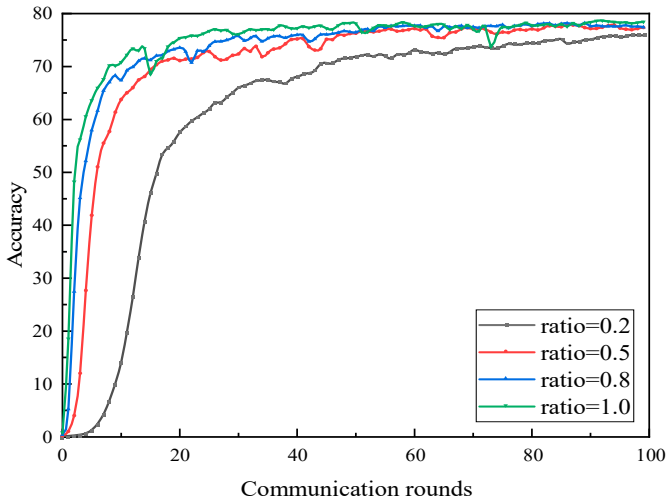
Table 3. Comparison of DP-FL methods on CIFAR-10 over two participated clients.

Schemes	Test Accuracy %		Communication Rounds
	IID	Non-IID (α = 15)	
DP-FedAvg [12]	82.68	81.12	200
f-DP [25]	80.37	78.84	200
Wu et al. [13]	81.25	79.05	170
ASPFL (ours)	83.57	82.43	120

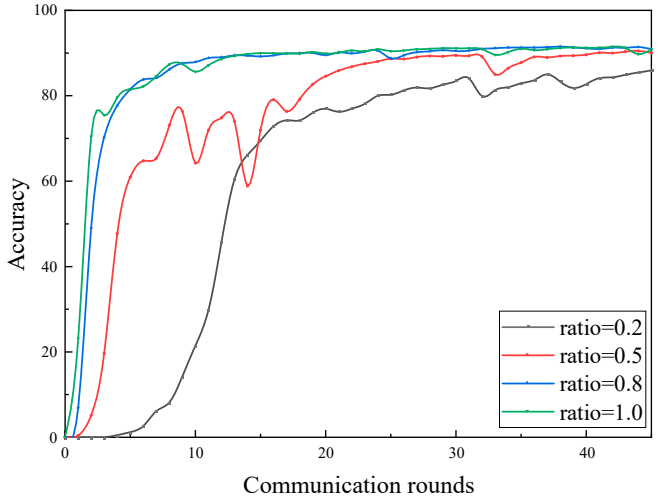
f To analyze the communication overhead of our proposed method, Table 3 also lists the number of communication rounds (CR) required for the model to reach convergence. Our method demonstrates a relatively low number of communication rounds (120), potentially offset by improved accuracy. Relative to DP-FedAvg [24] (200) and f-DP [25] (200), our proposed ASPFL reduces the number of CRs by 40% for local optimization. In summary, our method maintains relatively low communication overhead while achieving competitive accuracy. These enhancements collectively enable federated learning to improve efficiency and privacy protection while maintaining accuracy. Specifically, the introduction of adaptive Gaussian noise, along with the use of the Jensen-Shannon divergence as a metric for generating sparse matrices, contributes to more efficient model updates. This reduces the communication overhead, which is crucial in distributed learning settings where data is stored across multiple devices.

4.4. Impact on the Number of Participants

The following presents the results of the ASPFL-p system under different participant client numbers. We set the participated clients to be 2, 5, 8, and 10. Figure 6 depicts the accuracy curves corresponding to various ratios of participated clients (20%, 50%, 80%, and 100%) with non-IID Dirichlet (α = 0.5) on the CIFAR-10 and Fashion-MNIST datasets.



(**a**) on the CIFAR10



(**b**) on the Fashion_MNIST

Figure 6. Test accuracy over different ratios of participating clients with Dirichlet (α = 0.5).

As shown in Figure 6, it appears that increasing the number of participating clients leads to faster convergence and better final performance. On one hand, it attributes to the diversity and large amount of data because more clients mean a better representation of the underlying data distribution. Federated learning aggregates gradients from multiple clients, which helps the model learn from a wider variety of data points. This diversity can lead to better generalization and thus improved performance. On the other hand, with more clients contributing to the gradient aggregation, the resulting gradients are more stable and less noisy. This stability helps the model converge more quickly and accurately.

5. Conclusions

We present a communication-efficient federated learning framework with a privacy guarantee in this paper. Before transmitting model updates, a sparsity-pruning-based mechanism is employed based on the JS divergence of each client's data distribution.

When introducing privacy protection in model updates, we propose an adaptive Gaussian noise strategy to achieve (ε, δ)-DP. Based on experimental results in terms of accuracy and communication cost, we can conclude that our ASPFL scheme exhibits robust performance and competitive accuracy on the CIFAR-10 dataset. From another perspective, a multitude of experiments have demonstrated the portability of our proposed strategies to other domains. With an increase in data diversity and complex tasks, the methods proposed in this paper are expected to achieve an even better balance on accuracy and overhead in the future.

Author Contributions: Methodology, S.S.; software, S.S.; validation, S.S.; formal analysis, S.D.; investigation, Y.S.; writing—original draft preparation, S.S.; writing—review and editing, Y.Z.; supervision, Y.Z.; funding acquisition, Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (Grant No. U2032125), Shanghai Talent Development Fund (Grant No. E1322E1), Yongxin Zhu.

Data Availability Statement: The dataset used in this study is publicly available. The CIFAR10 dataset can be obtained from the following link: http://www.cs.toronto.edu/~kriz/cifar.html (accessed on 7 July 2024).

Conflicts of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2015**, arXiv:1409.1556. [CrossRef]
2. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis.* **2015**, *115*, 211–252. Available online: https://link.springer.com/article/10.1007/s11263-015-0816-y (accessed on 7 June 2024). [CrossRef]
3. Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv* **2019**, arXiv:1810.04805. Available online: http://arxiv.org/abs/1810.04805 (accessed on 28 May 2024).
4. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; Arcas, B.A.Y. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, PMLR, Lauderdale, FL, USA, 20–22 April 2017; pp. 1273–1282. Available online: https://proceedings.mlr.press/v54/mcmahan17a.html (accessed on 7 June 2024).
5. Yang, Q.; Liu, Y.; Chen, T.; Tong, Y. Federated Machine Learning: Concept and Applications. *arXiv* **2019**, arXiv:1902.04885. Available online: http://arxiv.org/abs/1902.04885 (accessed on 13 April 2024). [CrossRef]
6. Denil, M.; Shakibi, B.; Dinh, L.; Ranzato, M. Predicting Parameters in Deep Learning. *arXiv* **2014**. [CrossRef]
7. Michel, P.; Levy, O.; Neubig, G. Are Sixteen Heads Really Better than One? *arXiv* **2019**, arXiv:1905.10650. Available online: http://arxiv.org/abs/1905.10650 (accessed on 14 June 2024).
8. Konečný, J.; McMahan, H.B.; Yu, F.X.; Richtárik, P.; Suresh, A.T.; Bacon, D. Federated Learning: Strategies for Improving Communication Efficiency. *arXiv* **2017**, arXiv:1610.05492v2. Available online: https://arxiv.org/abs/1610.05492v2 (accessed on 8 June 2024).
9. Balle, B.; Cherubin, G.; Hayes, J. Reconstructing Training Data with Informed Adversaries. *arXiv* **2022**, arXiv:2201.04845. Available online: http://arxiv.org/abs/2201.04845 (accessed on 27 December 2023).
10. Wang, Z.; Song, M.; Zhang, Z.; Song, Y.; Wang, Q.; Qi, H. Beyond Inferring Class Representatives: User-Level Privacy Leakage From Federated Learning. In Proceedings of the IEEE INFOCOM 2019—IEEE Conference on Computer Communications, Paris, France, 29 April–2 May 2019; pp. 2512–2520. [CrossRef]
11. Wang, L.; Jia, R.; Song, D. D2P-Fed: Differentially Private Federated Learning with Efficient Communication. *arXiv* **2021**, arXiv:2006.13039. Available online: http://arxiv.org/abs/2006.13039 (accessed on 19 January 2024).
12. McMahan, H.B.; Ramage, D.; Talwar, K.; Zhang, L. Learning Differentially Private Recurrent Language Models. *arXiv* **2018**, arXiv:1710.06963. Available online: http://arxiv.org/abs/1710.06963 (accessed on 7 January 2024).
13. Wu, X.; Zhang, Y.; Shi, M.; Li, P.; Li, R.; Xiong, N.N. An adaptive federated learning scheme with differential privacy preserving. *Future Gener. Comput. Syst.* **2022**, *127*, 362–372. [CrossRef]
14. Shi, Y.; Liu, Y.; Wei, K.; Shen, L.; Wang, X.; Tao, D. Make Landscape Flatter in Differentially Private Federated Learning. In Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 17–24 June 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 24552–24562. [CrossRef]
15. Mireshghallah, F.; Backurs, A.; Inan, H.A.; Wutschitz, L.; Kulkarni, J. Differentially Private Model Compression. *arXiv* **2022**, arXiv:2206.01838. Available online: http://arxiv.org/abs/2206.01838 (accessed on 13 June 2024).

16. Lin, J. On the Interaction Between Differential Privacy and Gradient Compression in Deep Learning. *arXiv* **2022**, arXiv:2211.00734. Available online: http://arxiv.org/abs/2211.00734 (accessed on 13 June 2024).
17. Li, M.; Lai, L.; Suda, N.; Chandra, V.; Pan, D.Z. PrivyNet: A Flexible Framework for Privacy-Preserving Deep Neural Network Training. *arXiv* **2018**, arXiv:1709.06161. Available online: http://arxiv.org/abs/1709.06161 (accessed on 13 June 2024).
18. Wang, J.; Bao, W.; Sun, L.; Zhu, X.; Cao, B.; Yu, P.S. Private Model Compression via Knowledge Distillation. *arXiv* **2018**, arXiv:1811.05072. Available online: http://arxiv.org/abs/1811.05072 (accessed on 13 June 2024). [CrossRef]
19. Yang, J.; Chen, S.; Wang, G.; Wang, Z.; Jie, Z.; Arif, M. GFL-ALDPA: A gradient compression federated learning framework based on adaptive local differential privacy budget allocation. *Multimed. Tools Appl.* **2024**, *83*, 26349–26368. [CrossRef]
20. Konečný, J.; McMahan, H.B.; Ramage, D.; Richtárik, P. Federated Optimization: Distributed Machine Learning for On-Device Intelligence. *arXiv* **2016**, arXiv:1610.02527. [CrossRef]
21. Dwork, C.; McSherry, F.; Nissim, K.; Smith, A. Calibrating Noise to Sensitivity in Private Data Analysis. In *Theory of Cryptography*; Springer: Berlin/Heidelberg, Germany, 2006; Volume 3876.
22. Abadi, M.; Chu, A.; Goodfellow, I.; McMahan, H.B.; Mironov, I.; Talwar, K.; Zhang, L. Deep Learning with Differential Privacy. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; ACM: New York, NY, USA, 2016; pp. 308–318. [CrossRef]
23. Dwork, C.; Roth, A. The Algorithmic Foundations of Differential Privacy. *Found. Trends® Theor. Comput. Sci.* **2014**, *9*, 211–407. [CrossRef]
24. McMahan, H.B.; Andrew, G.; Erlingsson, U.; Chien, S.; Mironov, I.; Papernot, N.; Kairouz, P. A General Approach to Adding Differential Privacy to Iterative Training Procedures. *arXiv* **2019**, arXiv:1812.06210. Available online: http://arxiv.org/abs/1812.06210 (accessed on 22 January 2024).
25. Zheng, Q.; Chen, S.; Long, Q.; Su, W.J. Federated f-Differential Privacy. *arXiv* **2021**, arXiv:2102.11158. [CrossRef]
26. Kim, M.; Günlü, O.; Schaefer, R.F. Federated Learning with Local Differential Privacy: Trade-Offs Between Privacy, Utility, and Communication. In Proceedings of the ICASSP 2021—2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toronto, ON, Canada, 6–11 June 2021; pp. 2650–2654. [CrossRef]
27. Ling, X.; Fu, J.; Wang, K.; Liu, H.; Chen, Z. ALI-DPFL: Differentially Private Federated Learning with Adaptive Local Iterations. *arXiv* **2023**, arXiv:2308.10457. Available online: http://arxiv.org/abs/2308.10457 (accessed on 2 February 2024).
28. Jiang, Y.; Wang, S.; Valls, V.; Ko, B.J.; Lee, W.-H.; Leung, K.K.; Tassiulas, L. Model Pruning Enables Efficient Federated Learning on Edge Devices. *arXiv* **2022**, arXiv:1909.12326. Available online: http://arxiv.org/abs/1909.12326 (accessed on 6 January 2024). [CrossRef] [PubMed]
29. Zhu, Z.; Shi, Y.; Luo, J.; Wang, F.; Peng, C.; Fan, P.; Letaief, K.B. FedLP: Layer-Wise Pruning Mechanism for Communication-Computation Efficient Federated Learning. In Proceedings of the ICC 2023—IEEE International Conference on Communications, Rome, Italy, 28 May–1 June 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 1250–1255. [CrossRef]
30. Zhu, Z.; Shi, Y.; Xin, G.; Peng, C.; Fan, P.; Letaief, K.B. Towards Efficient Federated Learning: Layer-Wise Pruning-Quantization Scheme and Coding Design. *Entropy* **2023**, *25*, 1205. [CrossRef] [PubMed]
31. He, Y.; Xiao, L. Structured Pruning for Deep Convolutional Neural Networks: A Survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2024**, *46*, 2900–2919. [CrossRef] [PubMed]
32. Qian, Y.; Rao, L.; Ma, C.; Wei, K.; Ding, M.; Shi, L. Toward Efficient and Secure Object Detection With Sparse Federated Training Over Internet of Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2024**, 1–14. [CrossRef]
33. Collins, L.; Hassani, H.; Mokhtari, A.; Shakkottai, S. Exploiting Shared Representations for Personalized Federated Learning. *arXiv* **2023**, arXiv:2102.07078. [CrossRef]
34. Balle, B.; Barthe, G.; Gaboardi, M. Privacy Amplification by Subsampling: Tight Analyses via Couplings and Divergences. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2018. Available online: https://proceedings.neurips.cc/paper_files/paper/2018/hash/3b5020bb891119b9f5130f1fea9bd773-Abstract.html (accessed on 26 August 2024).
35. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. [CrossRef]