Cepe-FL: Communication-Efficient and Privacy-Enhanced Federated Learning Via Adaptive Compressive Sensing

Ye Liu, *Member, IEEE*, Shan Chang, *Member, IEEE*, and Yiqi Liu

***Abstract*—Model updates are exchanged between server(s) and participants in Federated Learning (FL), which can result in excessive delay, especially for large models. Existing communication-efficient FL approaches such as quantization and top-k sampling apply compression to gradients assuming that gradients are sparse and can tolerate small deviations. This can hardly be applied to down-link transmission. In this work, we employ compressive sensing on model parameters instead of gradients and propose a two-way adaptive compression scheme, Cepe-FL, which exploits dictionary learning to project non-sparse model parameters into sparse representations to ensure reconstruction accuracy. Cepe-FL supports joint model reconstruction with drastic reduction in computational complexity from $O(n)$ to $O(1)$. Cepe-FL adjusts the compression ratio adaptively according to the training loss, achieving the best trade-off between communication and model precision. Furthermore, it demonstrates efficacy in defending against membership inference attacks since only compressed models are exchanged. We conduct extensive experiments on three image classification tasks and compare with three communication-efficient approaches including FedPAQ, FedAvg and T-FedAvg. Cepe-FL presents the best performance in all tasks under IID and non-IID scenarios. We also implement white-box membership inference attacks, and the results show Cepe-FL can significantly suppress success ratio of inference in comparison with other approaches.**

***Index Terms*—Communication-efficient, compressive sensing, dictionary learning, federated learning, membership inference attacks.**

I. Introduction

FEDERATED learning (FL), a decentralized machine learning (ML) framework, enables multiple participants to train a global model collaboratively without exposing their individual private data. Within FL, updating the global model involves multiple rounds. During each iteration, a central server employs random selection to determine which participants will take part in the training process. The selected participants provide their local updates to the server, which then aggregates obtained updates to achieve a new global model. FL inherently preserves the data privacy of participants, in which learning is carried out in a distributed manner across a potentially large number of participants, and the server is responsible for aggregation.

Nevertheless, distributed FL training also gives rise to a series of challenges. One of the primary challenges is the excessive communication cost stemming from the transmission of model updates. Modern machine learning (ML) model architectures contain billions of parameters, particularly occupying gigabytes (GB) of space. It implies petabytes (PB) level communication cost before global model converging. Moreover, limited bandwidth and unreliable network conditions can lead to significant transmission delays, which seriously impacts the performance of FL. Another challenge is the risk of privacy leakage. Even though FL allows participants to keep their local data private, local updates are still vulnerable to privacy inferring attacks, e.g., membership inference attack. It is because local updates (either model parameters or gradients) may contain useful information of the corresponding training data that can be learnt by attackers and then utilized to infer information of interest. Thus, finding a communication-efficient and privacy-enhanced FL method is of great importance.

There generally exists two ways to reduce the communication costs of FL. First, there have been significant efforts on minimizing the communication in each iteration. In literature, up-streaming communication is reduced by gradient quantization [1], [2], [3]. The rationale behind this is that gradients signify the direction in which the model loss is minimized and can tolerate discrepancies to a certain extent. Nevertheless, gradient quantization can only achieve a relatively modest compression ratio. There is a trade-off between model convergence speed and compression ratio. Literature [4], [5], [6] exploits the sparsity of gradients and uploads only the top-k gradients, which consequently curtails the volume of traffic in each round. A. F. Aji et al. [5] demonstrate that retaining just 0.1% gradients allows the model with high precision. Several approaches [7], [8], [9] merge quantization with top-k sparsification, to further reduce communication overhead. The second way is to accelerate the convergence of the global model so as to reduce overall communication costs. For example, FedAvg, a SOTA FL algorithm [10], reduces the number of global models by increasing local training iterations. Participants update their local models for several times (with multiple batches) rather than once using only one

Manuscript received 2 September 2023; revised 15 November 2023; accepted 23 November 2023. Date of publication 20 May 2024; date of current version 4 September 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 61972081, in part by the Natural Science Foundation of Shanghai under Grant 22ZR1400200, and in part by the Fundamental Research Funds for the Central Universities under Grant 2232023Y-01. Recommended for acceptance by Special Issue On Federated Learning for Big Data Applications. *(Corresponding author: Shan Chang.)*

The authors are with the School of Computer Science & Technology, Donghua University, Shanghai 201620, China (e-mail: liuye@mail.dhu.edu.cn; changshan@dhu.edu.cn; liuyiqi@mail.dhu.edu.cn).

Digital Object Identifier 10.1109/TBDATA.2024.3403393

batch before uploading gradients. As a matter of fact, these two types of compression techniques are often combined to achieve the best compression performance [11], [12], [13].

The main disadvantage of the above approaches is that they are applied to gradients, which do not apply to down-stream compression, i.e., global model broadcasting. The reason is that different groups of participants serve as training workers in different iterations. This implies that, in each iteration, the version of the global model held by the dynamically joined participants is distinct. In order to maintain synchronization across all participants, during down-link communication, the server distributes the latest global model to all workers rather than incremental updates, i.e., gradients. T-FedAvg [13] compresses model parameters rather than gradients to support both up-link and down-link compression, however, which induces significant computational overhead on participants. Moreover, as the privacy vulnerability of FL caused by sharing model updates attracts more and more attention, it is desired that federated training procedures can also provide privacy protection for local updates. However, existing communication-efficient methods fail to provide resistance against privacy inference attacks. Both top-k and quantization preserve the most important features of updates, which can be utilized by attackers.

In this work, we devise a communication-efficient and privacy-enhanced FL method, Cepe-FL. Cepe-FL compresses model parameters instead of gradients by utilizing Compressive Sensing (CS), and is independent of the sparsity of model parameters. Cepe-FL facilitates two-way communication compression, resulting in a large reduction of the total amount of communication during federated training. The feasibility of Cepe-FL is that, according to CS, the number of samples required to recover a sparse signal can be far less than that required by the Shannon-Nyquist sampling theorem. The sparser the signal is, the better the reconstruction is. In other words, a signal might be reconstructed accurately from extremely incomplete measurements as long as it can be represented in a sparse way, i.e., mapped into a sparse domain [14], [15].

In Cepe-FL, it is essential to reconstruct the local model parameters with high precision, however, which is challenging, particularly under a large compression ratio. First, model parameters are high-dimensional and non-sparse. Finding the sparse representations of model parameters is non-trivial. Generally used transforms, e.g., Discrete Cosine Transform (DCT) and Discrete Fourier Transform (DFT), fail to guarantee low sparsity, and lead to poor reconstruction performance. Second, signal recovery algorithms, e.g., Orthogonal Matching Pursuit (OMP), are usually computationally intensive. In FL systems, especially under cross-device FL scenarios with massive participants, recovering compressed model parameters of individual participants produces excessive computational overhead on the server side, severely delaying model updating. Third, improving communication efficiency implies increasing the compression ratio and consequently decreasing recovery precision. Excessive compression ratios will result in huge recovery deviations, poor global model performance, and even damage model convergence. Last but not least, even when compressed local updates are transmitted, attackers can still recover the original ones (the lower the compression ratio, the better the reconstruction accuracy), jeopardizing data privacy. In general, it is extremely difficult to determine the optimal compression ratio during the entire federated training process to assure efficient communication and data privacy, as well as guarantee global model performance.

We propose three key techniques to tackle these challenges. First, we utilize dictionary learning (with a quasi-validation dataset on the server) to learn the representation basis (i.e., basis matrix), which can effectively map the model parameters to their sparse representations, rather than using a random generated basis. Second, we propose that the computational complexity of reconstruction on the server side can be reduced from $O(n)$ to $O(1)$ (where n is the number of participants involved in one iteration) by exploiting the linearity of compression. Then, we further reduce the computational cost of compression through layer-wise compression. Lastly, we design an algorithm to adjust the compression ratio adaptively, achieving the best trade-off between communication cost and model performance.

We claim that Cepe-FL is more resistant to privacy inference attacks due to the following two reasons. First, only a few of samples or measurements rather than the original local updates are uploaded. Second, attackers without the representation basis cannot recover the original local updates accurately. Notice that the representation basis is learnt from a quasi-validation dataset which is kept secret by the server.

We conduct extensive experiments to examine the performance of Cepe-FL. We verify Cepe-FL on three image classification tasks and compare it with FedAvg [10], FedPAQ [11] and T-FedAvg [13] under different data distributions. The experiment results indicate that Cepe-FL demonstrates superior performance compared to the other three methods in all settings. The results also support that representation basis can be learnt by a small quasi-validation dataset and is much more effective in signal reconstructing than general transforms. Finally, we investigate if Cepe-FL can provide stronger privacy protection against membership inference attacks. The experimental results show that the Cepe-FL can reduce the attack success rate by up to 20% under white-box passive inference attacks.

This paper is organized as follows. Section II presents related work. Preliminaries are described in Section III. Section IV presents the design of Cepe-FL. Section V demonstrates the experimental results. Section VI introduces the conclusion.

II. Related Work

A. Communication-Efficient FL

Many approaches have been designed to decrease the communication overhead during federated training. Generally speaking, these approaches can be divided into three classes: decreasing iteration, quantization, and gradient sparsification.

Decreasing Iteration: Suppress the number of update exchanges between the server and participants. B. McMahan et al. introduce a communication-efficient FL approach, FedAvg, the participants perform multiple iterations of local training before uploading their local updates to the server, reducing required communication rounds by a factor of $10\times$ to $100\times$.

However, the performance of FedAvg will significantly decrease on non-IID data [10]. T. Li et al. design a more versatile approach called FedProx. This approach allows each participant to determine the number of local training iterations dynamically, and demonstrates better performance on non-IID data compared with FedAvg [16]. To rectify the heterogeneity of local datasets in FedAvg, S. P. Karimireddy et al. propose Scaffold, which uses control variables (variance reduction) to achieve more accurate updates and faster convergence [17]. X. Yao et al. introduce MMD constraints in the training iteration of FL, which means using a dual stream model with MMD (maximum mean difference) constraints on each selected device, which can learn more general features from the global model, thereby reducing communication rounds without affecting model accuracy [18]. T. Lin et al. combine batch SGD and local SGD, proving that it can balance communication efficiency and model performance well [19]. J. Wang et al. adaptively adjust the number of local model updates, so in FL training, the running time can be minimized in the case of communication delays [20]. Y. Xiong et al. propose FedDM, a method based on iterative agent minimization. By transmitting more information and smaller synthetic data, communication rounds can be reduced, and model accuracy improved [21].

However, reducing the number of global model update iterations can only moderately reduce communication overhead. The reason is that it will significantly increase the number of local training iterations which not only implies much local computational and training time cost, but also leads to much slower convergence speed or even training collapse of the global model, due to the differences among local models, which are excessively iterated locally, will significantly increase, particularly when participants hold non-IID datasets.

Quantization: Each value of local gradients is converted from a 32-bit float number into a representation with fewer bits. J. Bernstein et al. introduce the signSGD method, which quantizes gradients into binary values, realizing a compression ratio of $32\times$ [1]. W. Wen et al. randomly quantize gradients into ternary values $0, 1, -1$, obtaining a compression ratio of $16\times$ [2]. D. Alistarh et al. quantize 32-bit floating-point values into 8-bit numbers, implying a compression ratio of $4\times$ [3]. Nevertheless, in the above study, the degree of quantification could not dynamically change throughout the entire FL training process. To this end, D. Jhunjhunwala et al. propose AdaQuantFL, an adaptive quantization algorithm that balances errors and communication bits, allowing the client to automatically adjust the quantization level of QSGD during the FL training process [22]. M. M. Amiri et al. propose LFL, a lossy FL method that quantifies global and client model parameters to further reduce communication costs [23]. S. Chen et al. propose FedHQ, which allocates different aggregation weights to clients by optimizing the convergence upper limit, which is suitable for heterogeneous FL scenarios where local clients have different quantization levels [24]. P. S. Bouzinis et al. quantize local model parameters before uplink transmission to minimize the total convergence time of FL [25].

Quantization-based approaches are commonly be used on compressing updates of gradients rather than model parameters, which implies they can only reduce uplink communication. Furthermore, quantization will result in large data precision loss, which may cause serious performance degradation in terms of convergence speed and model accuracy.

Sparsification: Only upload important gradients. A. F. Aji & K. Heafield argue that, during gradient uploading, 99% of the smallest gradients can be ignored without damaging final model accuracy [5]. N. Strom propose that only a small fraction of gradients whose absolute values are larger than a specific threshold are required to be exchanged *zero*, and gradient residuals are aggregated continuously to ensure that model weights with small gradients can eventually be updated. Nevertheless, determining an optimal threshold value poses a challenge [4]. Y. Lin et al. introduce Deep Gradient Compression (including gradient clipping, momentum correction, momentum factor masking, and training warm-up), which achieves a maximum compression ratio of $600\times$ on DeepSpeech [6]. A. Sahu et al. propose a hard threshold sparser for the top-k algorithm, which adaptively determines the sparsity k through a constant hard threshold. Consider the optimality throughout the training process rather than the optimality of each iteration [26]. P. Han et al. minimize the training time by adjusting the sparsity k to balance computation and communication [27]. M. K. Nori et al. characterizes learning errors through local updates and sparsity budgeting, and adaptively adjust these two components to generate fast FL (FFL) [28]. A. M. Abdelmoniem and M. Canini design an adaptive compression control mechanism. Combining network delay and compression control, can better balance training speed and model accuracy [29]. R. Jin et al. propose a channel-aware sparsity mechanism and studies the impact of sparsity and wireless channels on FL performance, achieving collaborative training of heterogeneous clients on wireless networks [30].

Same like quantization, most sparsification-based approaches only support upstream communication reduction. Meanwhile, in large FL systems with a huge number of participants, sparsification, i.e., top-k, may lead to significant decrease of model accuracy, and the convergence characteristics of federated training have not been studied yet. Literature [7] can be applied to both upstream and downstream communication and demonstrates satisfactory performance, however induces huge computational workload on the client side, which severely slows down the generation of local model update.

B. Membership Inference Attack and Defense

In FL, membership inference attack refers to using the updates exchanged between the participants and the server to infer whether a certain data sample is in the training dataset of a particular participant or not. L. Zhu et al. propose the concept of deep leakage from gradient in multi-node machine learning systems, and demonstrate the attack effectiveness [31]. J. Geiping et al. show that, in FL, the attacker can faithfully reconstruct a particular training sample from the knowledge of averaged gradients [32]. It is also proved that an internal attacker can infer the characteristics of other training workers according to the model parameters or gradients observed [33], [34]. M. Nasr et al. investigate how a white-box attacker launches passive and active membership inference attacks on federated systems [35]. X. He et al. propose ALGAN, which uses GAN to

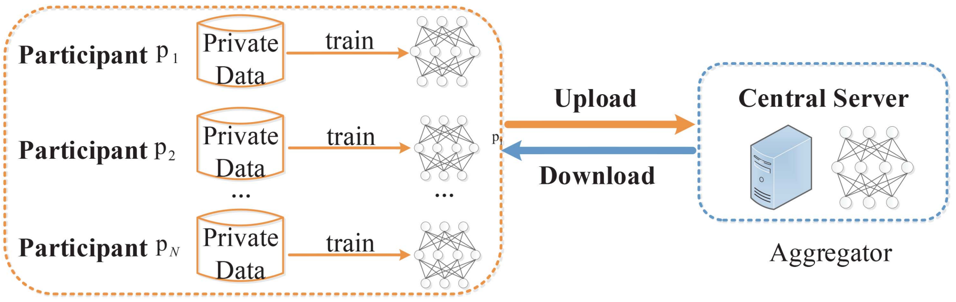

Fig. 1. Standard FL framework.

increase data diversity and applies active learning to label data generated by GAN. Due to the application of active learning in label data, membership inference attacks have high attack accuracy [36]. T. Nguyen et al. propose AMI, in which the server creates malicious parameters, and embeds them into the global model to effectively infer whether the target data sample is included in the private training data of the client [37]. Y. Gu et al. propose CS-MIA, a new membership inference based on predicted confidence sequences, which poses a more serious threat to the privacy of FL [38].

We now describe existing defenses against MIA. Intuitively, in existing research, differential privacy [39], [40] should be an effective method to make models robust to MIA. During the training process of the model, carefully design noise is injected into the descending gradient to ensure that the model meets the standards of differential privacy. However, this inevitably sacrifices the practicality of the model and increases the computational burden. In addition, there are other defense methods against MIA. R. Shokri et al. was the first to link membership inference to overfitting successfully, but this defense mechanism reduces the accuracy of the target model [41]. The defense mechanism proposed in the literature [42], [43] is based on adversarial training. J. Jia et al. intentionally add adversarial perturbations to the model output to interfere with adversarial reasoning and defend against black-box MIA. However, when only marking MIA is performed, it will fail [42]. M. Nasr et al. introduce adversarial regularization in the objective function to make the model robust to MIA. However, this approach often relies on strong assumptions about the attack model, making its effectiveness highly dependent on the similarity between the agent and the real attacker [43].

III. Preliminaries

A. System Model

We consider a FL framework that includes N decentralised participants $p_i, i \in \{1, \ldots, N\}$, as well as a central server $\mathcal{S}$. Each participant p_i has their own dataset $\mathcal{D}_i = \{x_{i,d}, y_{i,d}\}, d \in \{1, \ldots, |\mathcal{D}_i|\}$, where $x_{i,d}$ and $y_{i,d}$ denote the data samples and their corresponding labels, respectively. The purpose of the framework is to acquire a global model that can achieve robust testing data by aggregating the training outcomes (local models) of various participants (see Fig. 1).

TABLE I
List of Notations

Notations	Description
N	Number of total participants
p_i	Participant i
$\mathcal{D}_i = \{x_{i,d}, y_{i,d}\}$	Data collected by participant i
l	Number of select participants
$\mathbb{G}^t$	Global model at round t
$\mathbb{L}_i^t$	Local model of participant i at round t
$\mathcal{W}_{\mathbb{G}^t}$	Global model parameters at round t
$\mathcal{W}_{\mathbb{L}_i^t}$	Local model parameters of participant i at round t
$\mathcal{X}$	Quasi-validation set
Ψ	Sparse dictionary
Φ	Measurement matrix
$\mathcal{C}$	Compressed matrix
$\mathcal{S}$	Sparse coefficient matrix
δ	Compression ratio

In particular, the FL framework employs *FedAvg* as aggregation scheme. The federated training process involves multiple iterations. In iteration t, the server chooses $l\,(l \leqslant N)$ participants p_i^t to serve as workers, where $\lambda = l/N$ representing the participation ratio. Each training worker first downloads the current global model $\mathbb{G}^{t-1}$ from the server, and then independently applies Mini-Batch Gradient Descent (MBGD) several times to update $\mathbb{G}^{t-1}$ by using its own dataset and acquire the corresponding local update $\mathbb{L}_i^t$. Finally, each local update is sent to the server, where the global model $\mathbb{G}^t$ is obtained in (1), and the iteration t is complete.

$$\mathbb{G}^t = \sum_{i=1}^{l} \frac{|\mathcal{D}_i|}{\mathbf{D}^t} \mathbb{L}_i^t \tag{1}$$

where $\mathbf{D}^t$ represents the sum of training samples in round t, i.e., $\mathbf{D}^t = \sum_{i=1}^{l} |\mathcal{D}_i|$. The aforementioned procedures are executed iteratively until the learning process reaches convergence.

Additionally, we assume that the server holds a *quasi-validation set*, i.e., a set of data samples whose distribution is similar (if not the same) to that of the overall training samples (across training workers), denoted by $\{\mathcal{D}_1, .., \mathcal{D}_N\}$. Generally, the server is able to form such a *quasi-validation set* by collecting a few data samples in the same or similar domains; and labeling manually. Key symbols of significance for this paper are listed in Table I.

B. Compressive Sensing

Compressive Sensing (CS) [44], [45] implies "highly accurate reconstruction through highly incomplete linear measurement of signals." Under this sampling paradigm, the sampling rate no longer depends on the bandwidth of the signal but on two fundamental premises: sparse signal and incoherent sampling. The basic CS theory is composed of the sparse representation of the signal (with sparse basis), under-sampling (with the measurement matrix) and the sparse signal recovery.

Given a one-dimensional, real-valued discrete-time signal $\mathcal{W}$ with finite length, ready for sensing, it can be represented as a column vector of $n \times 1$ in the vector space $\mathbf{R}^n$. If $\mathcal{W}$ satisfies sparsity (whether in the original time domain or other transform domain), it is possible to capture almost all the information about $\mathcal{W}$ by using only m measurements, and m is much smaller than the dimension n of the original signal $\mathcal{W}$. In mathematics, the process of sensing $\mathcal{W}$ can be formulated as

$$\mathcal{Y} = \Phi \mathcal{W} \tag{2}$$

where Φ is the *measurement matrix* of $m \times n$ dimension, satisfying RIP (Restricted Isometry Property) [46].

Definition 1: (Signal Compression Ratio) refers to the relative reduction in signal representation size produced by certain signal compression algorithm. It can be calculated by the division of original size by corresponding compressed size:

$$CompressionRatio(\delta) = \frac{OriginalSize}{CompressedSize} \tag{3}$$

Following the application of compressive sensing, the signal $\mathcal{W}$ is decreased to $\mathcal{Y}$, with a compression ratio of n/m for $\mathcal{W}$.

IV. Sparsity Analysis on FL Model Parameters

Based on the concept of compressive sensing, the sparsity of model parameters is a prerequisite for accurate reconstruction. Therefore, in this section, we conduct the following experiments to check the sparsity of model parameters.

A. Models and Datasets

PyTorch's distributed library is applied to build an FL system which contains 100 participants. In each training iteration, a group of training workers will be selected from all participants by the server. A seven-layer Lenet5 is federated trained with MNIST [47] and F-MNIST [48], separately, and a six-layer CNN is trained with CIFAR-10 [49].In both MNIST and FMINST, 60000 images are training samples and 10000 images are testing samples, respectively. In CIFAR-10, there are 50000 training images and 10000 images for testing. In each FL task, the corresponding training datasets are divided into 100 subsets of equal size, and distributed to all participants. Table II shows the optimizer as well as hyper-parameters adopted in each training task.

TABLE II
Hyper-Parameters and Optimizers

Tasks	**Lenet5 on MNIST**	**Lenet5 on F-MNIST**	**CNN on CIFAR-10**
Batch size	64	128	128
Learning rate	0.01	0.1	0.001
Optimizer	SGD	SGD	Adam
Momentum	0.5	0.5	0.9
Round	100	100	200

B. Sparsity Factor (SF) of Model Parameters

In an $n \times m$ matrix, when the number of zero elements is much greater than the number of non-zero elements, and the arrangement of non-zero elements lacks regularity, it is referred to as a sparse matrix, otherwise, it is a dense matrix.

Definition 2. (Sparsity Factor): a commonly used metric of matrix sparsity denoted as SF, which is calculated by

$$SF = \frac{count(|x_i| > T)}{n \times m} \tag{4}$$

where the numerator is the number of elements in the matrix whose absolute value is greater than a given threshold T, and the denominator is the total number of elements in the matrix. Given T, small SF means high sparsity. In our experiments, we empirically set T as 0.001.

In each federated training task, we conduct 200 training iterations, and record the corresponding global model in each iteration. Then, for each model, we evaluate the SFs of the parameter metrics on each layer, and calculate the maximum, minimum, and average SFs, separately. From Fig. 3, we can see that the SFs of almost all layers are greater than 0.9, no matter the task. It is commonly considered that a matrix is sparse when $SF \leq 0.05$. Thus, we conclude that the model parameters are not sparse, and learning sparse representation of model parameters is crucial in CS.

C. Reconstruction Effectiveness of Common Sparse Basis

We use the commonly adopted orthogonal bases DCT as the sparse basis of the model parameters, and apply CS to the projected parameters (i.e., their sparse representations), to examine whether the recovered parameters lead to a precise global model or not. We also conduct 200 federated training iterations with parameter under-sampling on the training workers and sparse parameter recovery on the server, and test the precision of the global model after federated training.

Fig. 4 shows the testing precision of the global model under different compression ratios ($10\times$, $3\times$, $2\times$, $1\times$), where $1\times$ represents the parameters are uncompressed. High precision implies high recovery accuracy of parameters and efficacy of sparse representations.

In Fig. 4, it can be seen that, in all three tasks, as the compression ratio increases, the performance of the global model degrades significantly. For example, in the tasks of Lenet5 on F-MNIST and CNN on CIFAR-10, even a small compression ratio of $2\times$ causes a significant decay (around 10% and 20%, respectively) of testing precision. When the compression ratio is set to $10\times$, the global cannot even converge. The experimental result indicates that DCT is not an appropriate sparsifying basis.

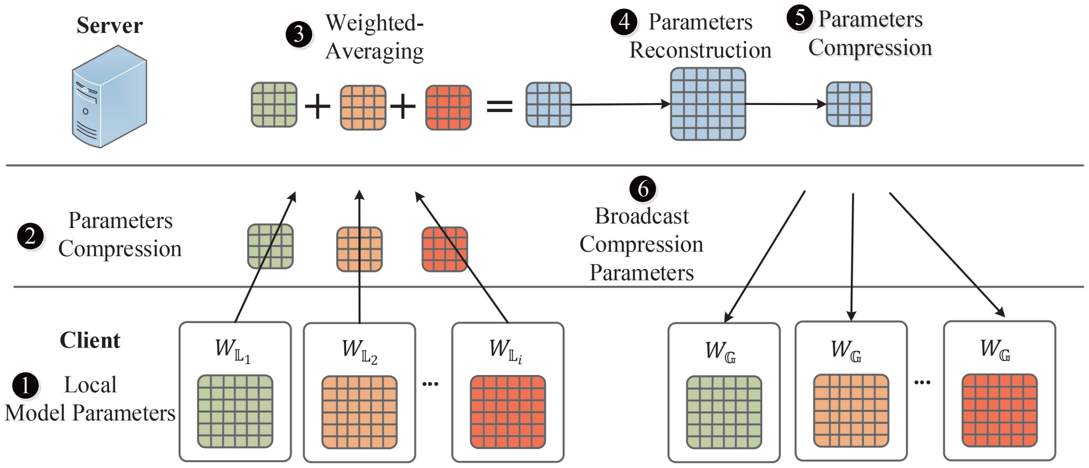

Fig. 2. The procedure performed by the server and workers to update the global model.

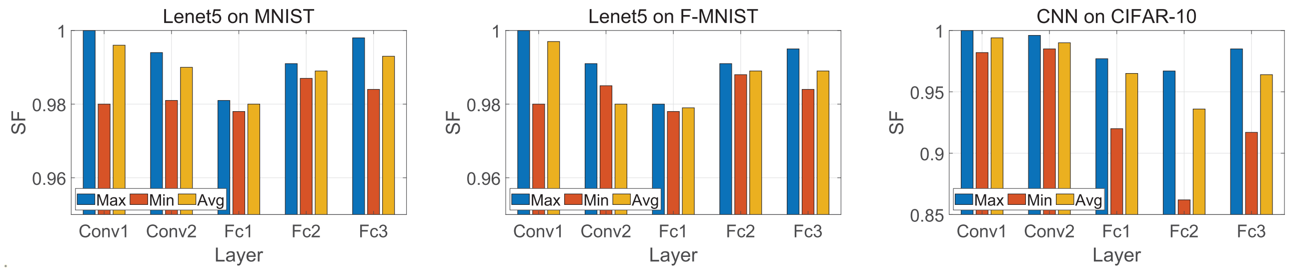

Fig. 3. The max. min. and avg. SFs of parameters on different layers.

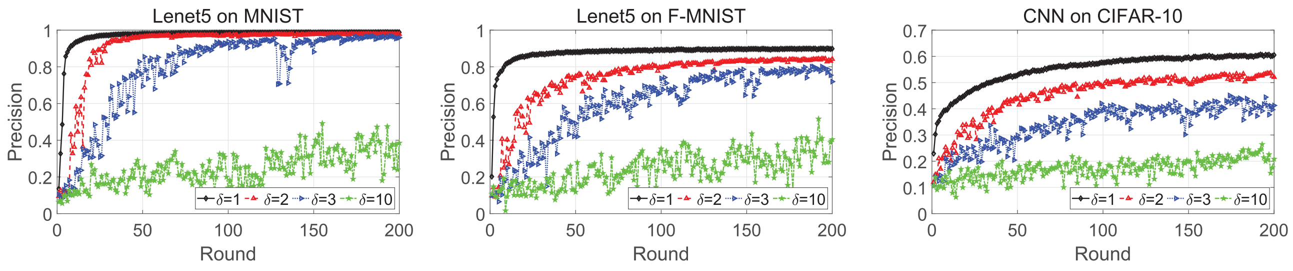

Fig. 4. Convergence trends under different compression ratios when model parameters are sparsified by DCT.

V. Design of Cepe-FL

A. Overview

In Cepe-FL, an initialization phase is prior to federated training. This process encompasses the following procedures (see Algorithm 1):

The central server first determines the architecture of the global model $\mathbb{G}^0$. The parameters of the model are n-dimensional and initialized as $\mathcal{W}_{\mathbb{G}^0}$. Second, the server constructs a Gaussian random matrix of $n \times n$ as a measurement matrix Φ, and the quasi-validation set $\mathcal{X}$ which will be used to learn a sparse dictionary Ψ of $n \times k$ $(k > n)$. Ψ will be employed to project the model parameters exchanged between the server and workers into sparse representations (see in Section V-B).

Each worker obtains Φ from the server to create a dynamic measurement matrix during each training round.

Next, in order to get an updated model $\mathbb{G}^1$, the server and workers execute the following process (see Fig. 2):

First, the server determines the compression ratio δ (see in Section V-C), which indicates that the compression version of $\mathcal{W}_{\mathbb{G}^0}$ has a dimensionality of m $(m = n/\delta)$, and broadcasts δ to all workers.

Second, following the acquisition of $\mathbb{G}^0$, each worker p_i executes the MBGD iterative, utilizing its private dataset $\mathcal{D}_i$, until the attainment of the local model $\mathbb{L}_i^1$, and converts $\mathcal{W}_{\mathbb{L}_i^1}$ into a column vector with dimensions $n \times 1$, and regards $\mathcal{W}_{\mathbb{L}_i^1}$ as

Algorithm 1: Cepe-FL.

```
Input: Global model $\mathcal{W}_{\mathbb{G}^0}$, compression ratio $\delta^0$,
  measurement matrix $\Phi$ and quasi-validation set $\mathcal{X}$.
1:  Init: Learn a sparse dictionary $\Psi$ using a
    quasi-validation set $\mathcal{X}$, broadcast $\mathcal{W}_{\mathbb{G}^0}$, $\delta^0$ and $\Phi$ to
    clients $p_i$, $i\epsilon\{1,2,..,N\}$, assign each client a unique
    dataset $D_i$.
2:  function ServerExecution
3:    for round $t$ in range $(T)$ do
4:      for $p_i \in l = \{1,2,\ldots,|\lambda N|\}$ do
5:        $\mathcal{C}_{\mathbb{L}_i^t} \leftarrow$ ClientUpdateModel $(p_i, \mathcal{W}_{\mathbb{G}^t}, \delta^t)$
6:      end for
7:      $\mathcal{C}_{\mathbb{G}^{t+1}} \leftarrow$ Aggeration $(\mathcal{C}_{\mathbb{L}_1^t}, \mathcal{C}_{\mathbb{L}_2^t}, \ldots, \mathcal{C}_{\mathbb{L}_N^t})$
8:      $\mathcal{W}_{\mathbb{G}^{t+1}} \leftarrow OMP(\mathcal{C}_{\mathbb{G}^{t+1}})$
9:      $\delta^{t+1} \leftarrow$ according to Algorithm 2
10:   end for
11:   broadcast $(\mathcal{W}_{\mathbb{G}^{t+1}}, \delta^{t+1})$ to all clients
12: end Function
13:
14: function ClientUpdateModel$(p_i, \mathcal{W}_{\mathbb{G}^t}, \delta^t)$
15:   $\mathcal{W}_{\mathbb{L}_i^t} \leftarrow \mathcal{W}_{\mathbb{G}^t}$
16:   $\Phi_\delta^{t+1} \leftarrow$ extract from $\Phi$ according to $\delta^t$
17:   $\beta \leftarrow$ (split $D_i$ into batches of size $B$)
18:   for local epoch $e$ in range $(E)$ do
19:     for batch $b \in \beta$ do
20:       $\mathcal{W}_{\mathbb{L}_i^{t+1}} \leftarrow \mathcal{W}_{\mathbb{L}_i^t} - \eta\nabla L(\mathcal{W}_{\mathbb{L}_i^t}; b)$
21:     end for
22:   end for
23:   $\mathcal{C}_{\mathbb{L}_i^{t+1}} = \Phi_\delta^{t+1} \times \mathcal{W}_{\mathbb{L}_i^{t+1}}$
24:   upload $\mathcal{C}_{\mathbb{L}_i^{t+1}}$ to server
25: end Function
```

the signal that needs to be compressed. Then, using the selection method, such as selecting the first m rows, the m rows in Φ will be selected to create a runtime matrix Φ_δ, which will be utilized to compress $\mathcal{W}_{\mathbb{L}_i^0}$ based on (2), i.e., $\mathcal{C}_{\mathbb{L}_i^1} = \Phi_\delta \mathcal{W}_{\mathbb{L}_i^1}$, and the compressed matrix $\mathcal{C}_{\mathbb{L}_i^1}$ is sent to the server.

Third, the server uses a reconstruction algorithm like OMP [50] to reconstruct the $\mathcal{W}_{\mathbb{L}_i^1}$ after receiving $\mathcal{C}_{\mathbb{L}_i^1}$, and the weighted-averaging aggregation mechanism is applied, resulting in $\mathbb{G}^1$. Nevertheless, the cost of reconstructing local models one by one on the server side is too high, so we introduce a joint reconstruction strategy wherein the server only requires one reconstruction process during each iteration (see in Section V-D1).

Obviously, downlink stream compression in Cepe-FL is easy, wherein the server utilizes Φ_δ to compress $\mathbb{G}^1$, while the workers are able to reconstruct $\mathbb{G}^1$ by employing $\Phi\Psi$, which may be acquired from the server during the initialization phase.

B. Learning Sparse Dictionary Ψ

Based on the compressive sensing theory [14], [15], one of the prerequisites for accurate reconstruction of model parameters (i.e., weights) $\mathcal{W}_{\mathbb{L}_i^t}$ is that they must be sparse. Given that the $\mathcal{W}_{\mathbb{L}_i^t}$ are not sparse in themselves, therefore $\mathcal{W}_{\mathbb{L}_i^t}$ needs to be projected to the sparse domain. To achieve this, we use a sparse representation basis denoted as Ψ, called sparse dictionary, make $\Psi\mathcal{S}_{\mathbb{L}_i^t} = \mathcal{W}_{\mathbb{L}_i^t}$, satisfying that $\mathcal{S}_{\mathbb{L}_i^t}$ is a s-sparse vector, representing projection coefficients of $\mathcal{W}_{\mathbb{L}_i^t}$ on Ψ.

Nevertheless, directly using commonly used predefined sparse dictionaries on different models, such as local Fourier basis and wavelet basis, cannot make model parameters sparse. Furthermore, during federated training, the properties of model parameters change over many iterations, so finding a dictionary that is generally valid for all models is a challenging task.

To discover a sparse representation of $\mathcal{W}_{\mathbb{L}_i^t}$, we employ a dictionary learning approach (see Fig. 5). One of the basic principles of dictionary learning is that a dictionary must be learned from input data. Nevertheless, during the learning process, the server lacks information regarding $\mathcal{W}_{\mathbb{L}_i^t}$. Hence, the server employs a small quasi-validation set $\mathcal{X}$ in order to iteratively train the global model $\mathbb{G}$ until convergence or reaching the predetermined number of iterations, and evenly chosen ζ intermediate models $\mathbb{G}^t$ acquired from each iteration of training. Subsequently, in order to acquire knowledge of Ψ, the server utilizes the parameters of ζ from previously recorded intermediate models, i.e., $\mathcal{W}'_{\mathbb{G}^t}$, as input for executing the K-SVD method. The method iteratively alternates between the sparse representation of $\mathcal{W}'_{\mathbb{G}^t}$, i.e., $\mathcal{S}'_{\mathbb{G}^t}$ according to the present dictionary, and updating the dictionary to more accurately fit the input [51].

Since the data distribution of $\mathcal{X}$ is like the actual sample distribution of workers, we apply $\mathcal{W}'_{\mathbb{G}^t}$ to learn Ψ. This implies that the model trained using dataset $\mathcal{X}$ exhibits similarity to models trained using datasets from multiple participants, denoted as $\{\mathcal{D}_1, .., \mathcal{D}_N\}$ (assuming the model structure and parameters remain consistent). Hence, it is justifiable to use the set of $\mathcal{W}'_{\mathbb{G}^t}$ to learn Ψ. The implementation steps of the sparse representation model parameter method based on the K-SVD dictionary learning algorithm are as follows:

- *Step 1:* Select the quasi-validation set $\mathcal{X}$ as the training data, train the global model, and select A model parameters to form a model parameter sample set matrix $\mathcal{W}''_{\mathbb{G}} \in R^{n\times A}$.
- *Step 2:* Determine relevant parameters, including the length n, the number k of atoms in the initial dictionary, and the number of iterations of the K-SVD algorithm.
- *Step 3:* Based on the initial dictionary atom length n, k atoms are randomly selected to form the initial dictionary $\Psi \in R^{n\times k}$, and then the OMP algorithm is used to obtain the sparse coefficient matrix $\mathcal{S} \in R^{k\times A}$ under the initial dictionary of model parameters, i.e., $\mathcal{W}''_{\mathbb{G}} = \Psi\mathcal{S}$.
- *Step 4:* According to the K-SVD algorithm, the initial dictionary atoms are continuously updated. If the number of iterations is reached, the update is stopped in order to construct the optimal over-complete dictionary Ψ that conforms to the characteristics of the model parameters.

C. Determining Compression Ratio δ

In general, a balance exists among the compression ratio, communication overhead, and model precision. As the compression ratio δ increases, the volume of data traffic decreases, but

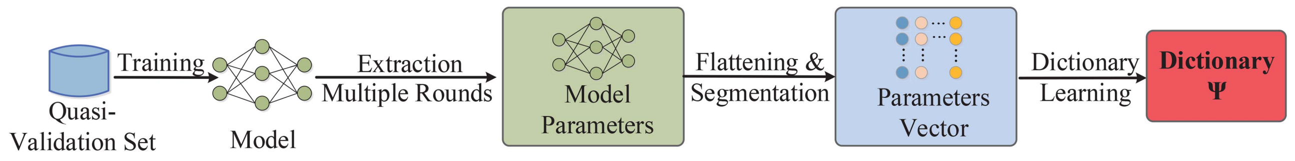

Fig. 5. The server learns a sparse dictionary by using the quasi-validation set.

this reduction comes at the expense of decreased reconstruction precision. It is important to underline that, in Cepe-FL, the utilization of a constant value δ in federated training does not result in optimal performance. This phenomenon may be attributed to the large variation of model parameters in the early phases of FL, which consequently allows for a comparatively high level of tolerance towards reconstruction errors. Nevertheless, as the training enters the later period, the model gradually tends to converge, and the model parameters can be adjusted minimally only through fine-tuning. The large reconstruction error hinders the continuous improvement of the precision of the reconstructed model. The rate of convergence may decrease substantially, and the number of communications may increase, resulting in an increase in communication costs. Furthermore, it leads to a decrease in model precision and the inability to successfully train the model. Hence, in each round of training, a suitable δ needs to be selected in order to minimize the overall communication expenses while avoiding substantial deterioration in accuracy.

We propose a new method for adjusting the compression ratio in an adaptive manner. Not only does it guarantee that the final model is nearly as accurate as FedAvg, but it also minimizes the communication overhead. The fundamental concept is to use a larger δ during the initial training phase, and then gradually decrease the δ based on the global model loss during the training process. The reason for associating adjustment δ with model loss is that there is an inverse correlation between the reduction of model loss and the improvement of model convergence. The process of adjusting δ by the server is as follows (see Algorithm 2):

- *Step 1:* Relevant parameters need to be determined, including the maximum number of training rounds, denoted as θ, in which a constant compression ratio is maintained. Additionally, the initial compression ratio, denoted as δ^0, is to be determined for use in the previous θ training rounds. Furthermore, the number of continuous rounds, denoted as ϑ, and a scaling factor, denoted as ϵ, are marginally less than 1.
- *Step 2:* After one round of training, save the loss of the global model $\mathbb{G}^0$, represented as ℓ^0.
- *Step 3:* In round t $(t > \theta)$, the server computes the average loss of the last θ rounds, represented as ℓ^t. Additionally, the server computes $\gamma = \ell^t/\ell^0$. It can be observed that ℓ^t reduces with t, and is non-negative, which means $0 \leqslant \gamma \leqslant 1$. The new compression ratio is set by the server to $\delta^t = \gamma \cdot \delta^0$.
- *Step 4:* If the server sees that the model loss remains constant for ϑ rounds, it indicates that the most recent model has converged or has significant reconstruction errors. In response, the server decreases δ^t to $\epsilon \cdot \delta^t$, to further enhance the precision of the model.

Algorithm 2: Adaptively Determine the Compression Ratio.

```
Input: Initial compression ratio $\delta^0$, maximum number of
 training rounds $\theta$, continuous rounds $\vartheta$, scaling factor $\epsilon$.
Output: Determine the compression ratio $\delta^t$ at round $t$.
    function ServerExecution
2:    for round $t$ in range $(T)$ do
        if $(t \leqslant \theta)$ then
4:        Record $\ell^0 \leftarrow$ the loss of the global model $\mathbb{G}^0$
          after one round training.
          Set $\delta^t = \delta^0$
6:      else
          Calculates $\ell^t \leftarrow$ the average loss of the last $\theta$
          rounds.
8:        Calculates $\gamma = \ell^t/\ell^0$, where $0 \leqslant \gamma \leqslant 1$.
          Set $\delta^t = \gamma \cdot \delta^0$.
10:     if (model loss is unchanged for $\vartheta$ consecutive
        rounds) then
          Set $\delta^t = \epsilon \cdot \delta^t$.
12:   end for
    end Function
```

D. Reducing Computational Overhead of Reconstruction

1) Jointly Recovering Model Parameters: In general, reconstruction methods tend to require significant computational resources. For instance, the complexity of the calculation required by the OMP reconstruction method is $O(mns)$ [50]. Therefore, in a round of training involving l workers, the computational overhead of reconstructing all local models is $l \cdot O(mns)$. In a large FL framework, it is important to decrease the complexity of computation associated with the reconstruction process. Thankfully, we discovered that reducing it from $l \cdot O(mns)$ to $O(mns)$ is feasible. That is to say, the server jointly reconstructs model parameters every time an update is made to the global model, regardless of the number of local models.

Theorem (Linearity of Compression): Given two distinct signal vectors, $\mathcal{W}_j$ and $\mathcal{W}_i$, of size $n \times 1$; an arbitrary measurement matrix Φ with dimensions $m \times n$, as well as constants a_j and a_i,

$$a_j \cdot \Phi\mathcal{W}_j + a_i \cdot \Phi\mathcal{W}_i = \Phi \cdot (a_j \cdot \mathcal{W}_j + a_i \cdot \mathcal{W}_i) \tag{5}$$

Equation (5) shows that the server can conduct weighted averaging before reconstructing the global model. Based on (1), we can establish the following relationship:

$$\mathcal{W}_{\mathbb{G}^t} = \sum_{i=1}^{l} \omega_i \cdot \mathcal{W}_{\mathbb{L}_i^t} \tag{6}$$

where $\omega_i = |\mathcal{D}_i|/\mathbf{D}^t$, thus we can conclude

$$\sum_{i=1}^{l} \omega_i \cdot \mathcal{C}_{\mathbb{L}_i^t} = \Phi \cdot \sum_{i=1}^{l} \omega_i \cdot \mathcal{W}_{\mathbb{L}_i^t} = \Phi \cdot \mathcal{W}_{\mathbb{G}^t} \tag{7}$$

Therefore, we first aggregate the local models, and then $\mathcal{W}_{\mathbb{G}^t}$ is reconstructed using the aggregated (compressed) local models.

Moreover, the method of joint reconstruction presents two obvious benefits. First, the reconstruction error will only occur once during each update. Second, since there are numerous workers training the global model, it is more likely that the distribution of training samples will be identical to that of the quasi-validation set. Therefore, even if workers possess non-IID data, a low reconstruction error can still be ensured.

2) Layer-Wise Compression: It is worth considering that workers typically refer to low-end equipment, and it is common for the number of model parameters to be significantly large. To decrease the compression overhead, we employ layer-wise compression. For instance, we consider a model with κ-layers, where the parameters of each layer are n-dimensional. To maintain generality, we make the assumption that all layers have the same size, and the compression ratio is denoted as $\delta = m/n$, which signifies that the measurement matrix Φ has dimensions of $m \times n$. Then the computational complexity of compressing the entire model is $O(n \cdot m)$. On the contrary, if we choose a layer-by-layer compression strategy, the dimensions of Φ are changed to $\frac{m}{\kappa} \times \frac{n}{\kappa}$, so the expense of compression could be diminished to $\kappa \cdot O(\frac{m \cdot n}{\kappa^2})$ (i.e., $O(\frac{m \cdot n}{\kappa})$). Therefore, it can be inferred that the implementation of layer-wise compression results in a reduction of the computational overhead to $1/\kappa$ compared to its original overhead.

VI. Security Analysis

We consider an attacker who aims at launching a membership inference attack successfully, i.e., inferring whether a data sample is in the training dataset of a target training worker v (i.e., victim participant). There are two types of attackers, i.e., *outsider* and *insider*. An outsider is someone who can eavesdrop the communications between the victim and the server, thus can obtain the compressed model parameters, i.e., $\mathcal{C}_v$, transferred between them. An inside attacker refers to a curious participant who is selected as a training worker in the FL task. The insider can obtain not only the compressed model parameters of its victim, but also the measurement matrix Φ used to compress the original model parameters as well.

We emphasize that the essential information needed for launching a local membership inference attack on the victim v is the up-link local model parameters $\mathcal{W}_{\mathbb{L}_v}$ [35], which is however compressed by using Φ. As we mentioned before, assume that $\mathcal{W}_{\mathbb{L}_v}$ and Φ are $n \times 1$ and $m \times n$ matrices, respectively. Then $\mathcal{C}_{\mathbb{L}_v}$ is a vector of $m \times 1$ dimensions calculated by

$$\mathcal{C}_{\mathbb{L}_v} = \Phi \mathcal{W}_{\mathbb{L}_v} \tag{8}$$

Accordingly, to recover $\mathcal{W}_{\mathbb{L}_v}$, the attacker should solve the above equation system which is however underdetermined (containing n variables in m equations), since $m \ll n$. Consequently, it is hardly to recover $\mathcal{W}_{\mathbb{L}_v}$, even with Φ.

Although, according to the CS theory, Φ satisfies RIP, to reconstructing $\mathcal{W}_{\mathbb{L}_v}$ properly requires the knowledge of the sparse base Ψ of $\mathcal{W}_{\mathbb{L}_v}$, held by the server only (who is assumed to be honest). In this case, efforts made by utilizing commonly used sparse bases, e.g., DCT and DFT etc., are proven to have poor reconstruction accuracy. As a result, both kinds of attackers cannot launch membership inference attacks effectively without Ψ, due to their low performance on reconstructing $\mathcal{W}_{\mathbb{L}_v}$.

VII. Experiment

A. Methodology

We conduct empirical validation on our Cepe-FL approach with three image classification tasks explained in Section IV-A. We compare the performance of Cepe-FL with three SOTA methods, i.e., FedAvg [10], FedPAQ [11] and T-FedAvg [13], on the precision of final models and overall communication overhead in FL. We investigate the influence of various FL settings on the performance of Cepe-FL. Furthermore, we demonstrate the effectiveness of Cepe-Fl to defend against membership inference attacks.

Comparisons: We conduct a comparative analysis between Cepe-FL and the following methods:

- *FedAvg:* most commonly used communication-efficient aggregation technique in FL, communication is limited by iterating local update multiple times before averaging [10].
- *FedPAQ:* integrates federated averaging and quantization techniques [11], obtaining a compression ratio of $4\times$ in each iteration by quantizing 32-bit values to 8-bit.
- *T-FedAvg:* a quantitative technique in FL [13]. In this method, the 32-bit float values are reduced to 2-bit, resulting in a compression ratio of $16\times$ in each iteration.

Notice that, in FedPAQ and T-FedAvg, the total number of training iterations may be increased due to loss of precision.Moreover, in Cepe-FL, the size of the first convolutional layers of both two models is very small, thus we do not compress them.

B. Reconstruction Effectiveness of Dictionary Learning

We examine the effectiveness of the dictionary learning algorithm adopted in our Cepe-FL (i.e., K-SVD) on model parameter reconstruction, by calculating the Mean Square Error (MSE) [52], which is defined as

$$MSE = \frac{1}{L} \sum_{l=1}^{L} \frac{||x_l - \hat{x}_l||_F^2}{N} \tag{9}$$

where x_l represents the original model parameters and $\hat{x}_l$ represents the reconstructed model parameters.

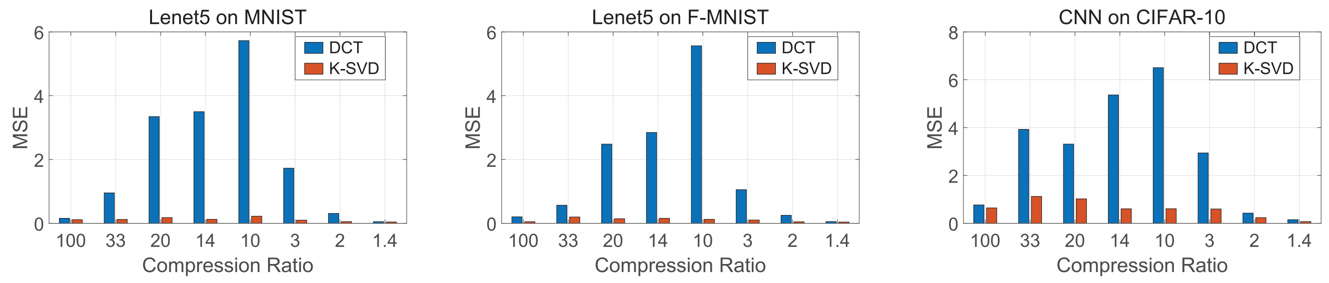

Fig. 6. MSEs of reconstructed model parameters vs. compression ratios.

TABLE III
KEY PARAMETERS IN K-SVD

Models	Lenet5 on MNIST	Lenet5 on F-MNIST	CNN on CIFAR-10
θ	15	5	2
δ^0	100	100	100
ϵ	0.9	0.9	0.9
size of $\mathcal{X}$	1000	100	500

In every FL task, we first conduct the global model training without CS for 200 iterations, and obtain the global model. Second, we apply CS to the local model updates each iteration, and record the global model after 200 iterations. Then we check the MSEs of the latter (reconstructed) model to the front model. The result is shown in Fig. 6. It can be seen that across all tasks and under different compression ratios, the MSEs of reconstructed model parameters when using K-SVD-based sparsifying basis are always much smaller than using DCT. Key parameters in K-SVD are illustrated in Table III.

C. Performance Comparison

In this experiment, the training data was divided among the workers in equal portions. Each training round has a worker participation rate of 0.1. We compare the communication overheads of four different approaches as well as their testing precision. The study involves both IID data, where each worker possesses all classes in the training data, and non-IID data, where each worker possesses 50% and 20% of the classes in the training data, respectively. After training for 100 rounds, the test precision and the associated upstream and downstream communication overheads for various methods are displayed in Table IV. Note that FedAvg and FedPAQ do not do downline compression. It is evident that Cepe-FL demonstrates similar levels of precision to FedAvg across all three methods while achieving substantial reductions in communication overheads of 76.2%, 39.7%, and 71.6%, respectively. The test precision of various methods is presented in Table V following the completion of a specified number of training rounds (Lenet5: 100 rounds, CNN: 200 rounds). It should be noted that the impact of data distributions on communication overheads remains constant when the number of training rounds is predetermined. Consequently, these effects are not included in the table. It is evident that the test precision of Cepe-FL exhibits greater proximity to that of FedAvg in comparison to the other two methods across all tasks.

The convergence curves are depicted in Fig. 7 (IID), Fig. 8 (non-IID, 50% classes) and Fig. 9 (non-IID, 20% classes) for various data distributions, depicting the mean quantity of bits required for workers to upload in order to attain specific levels of testing precision. Based on the obtained data, it can be concluded that our Cepe-FL method has superior performance compared to the other three methods across all scenarios. For the Lenet5 on MNIST task, no matter how the data is distributed, the ultimate communication overheads incurred by Cepe-FL are merely around one-tenth, one-third, and one-fourth of those associated with FedAvg, FedPAQ, and T-FedAvg, correspondingly. For the Lenet5 on F-MNIST task, it has been observed that when dealing with non-IID data, both the Cepe-FL and FedAvg methods are able to preserve the final test precision at levels comparable to those achieved on IID data. However, the FedPAQ and T-FedAvg methods experience variable degrees of decline in their final test precision. As the dissimilarities across the data distributions of workers increase, there is a corresponding decrease in precision. For the CNN on CIFAR10 task, despite the drop in precision of all methods on non-IID data, Cepe-FL consistently demonstrates ultimate precision that is comparable to that of FedAvg. However, the other two methods have much less precision in the end than FedAvg, for instance, the ultimate precision of FedAvg is approximately 55%, whereas the precision of T-FedAvg is 40% when applied to IID data. Moreover, Cepe-FL effectively reduces communication overheads by 35%-50% compared to FedAvg.

D. Influence of the Participation Ratio λ

In this subsection, we examine the impact of the parameter λ on the performance of the Cepe-FL. The experiments are carried out utilizing data from the IID. During each training round, we randomly choose $\lambda \cdot 100$ workers based on the λ given. The test precision acquired by Cepe-FL during training with varied participation ratios λ $(0.1, 0.3, 0.5, 0.7)$ are depicted in Fig. 10(a). We discover that Cepe-FL is not overly sensitive to variations in the value of λ. Even though Cepe-FL demonstrates a significantly slower learning time when using a little $\lambda = 0.1$, the overall test precision remains the same regardless of the λ used. This suggests that it is feasible to reduce the number of participants in FL tasks, resulting in significant savings in communication overheads. In addition, when λ grows, the performance fluctuations experience a minor reduction in their magnitude.

TABLE IV
TEST PRECISION AND COMMUNICATION OVERHEAD OF DIFFERENT METHODS WHEN TRAINED ON IID DATA

Tasks	Lenet5 on MNIST		Lenet5 on Fashion-MNIST		CNN on CIFAR-10	
	Precision	Communication overheads (KB) Upload/Download	Precision	Communication overheads (KB) Upload/Download	Precision	Communication overheads(KB) Upload/Download
FedAvg	0.985	24012/ 24012	0.893	24012/24012	0.576	24012/24012
FedPAQ	0.982	6003/ 24012	0.853	6100/4082	0.491	6100/7203
T-FedAvg	0.908	1864/1864	0.793	1864/1864	0.375	1900/1900
Cepe-FL	0.980	5712/5712	0.892	14472/14472	0.535	6828/6828

TABLE V
TEST PRECISION OF DIFFERENT METHODS WHEN TRAINED ON NON-IID DATA

Tasks	Lenet5 on MNIST		Lenet5 on F-MNIST		CNN on CIFAR-10	
	20%	50%	20%	50%	20%	50%
FedAvg	0.957	0.977	0.809	0.875	0.367	0.495
FedPAQ	0.924	0.975	0.766	0.833	0.244	0.381
T-FedAvg	0.643	0.863	0.432	0.639	0.145	0.330
Cepe-FL	0.956	0.975	0.788	0.868	0.287	0.445

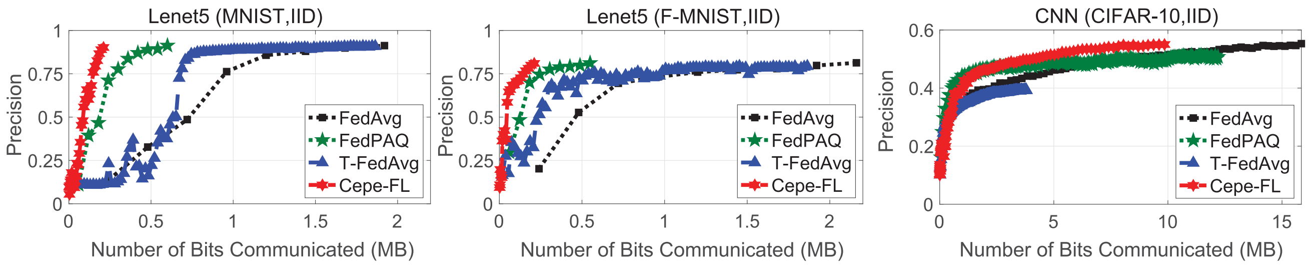

Fig. 7. Convergence trends and communication overheads with IID data.

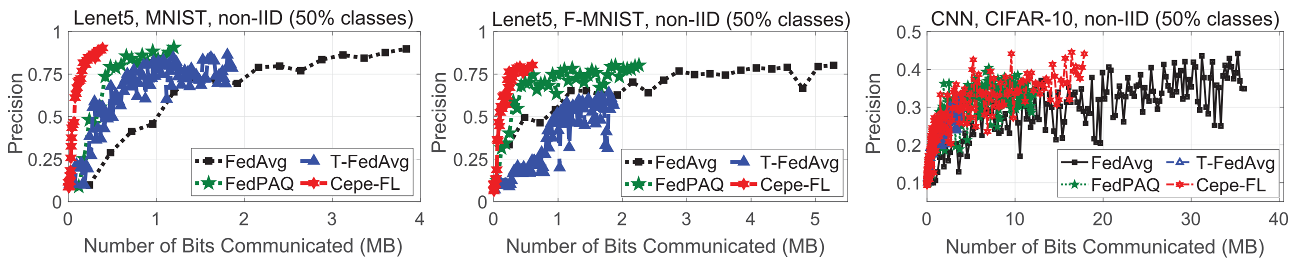

Fig. 8. Convergence trends and communication overheads with non-IID data (50% classes).

E. Influence of Unbalanced Datasets Among Participants

All of the above experiments were done with evenly split data sets, so that each worker had the same number of samples. In this subsection, we aim to assess the efficacy of the Cepe-FL in addressing the issue of data size imbalance [7]. The degree of data size imbalance is described using the formula

$$\mu = \frac{median\{\mathcal{S}_N\}}{\max\{\mathcal{S}_N\}} \tag{10}$$

where $\mathcal{S}_N = \{|\mathcal{D}_1|, .., |\mathcal{D}_N|\}$. When $\mu = 0.1$ indicates that the majority of the samples are concentrated in a limited number of workers, whereas $\mu = 1$ indicates an equal distribution of samples among all workers. We consider μ values of 0.1, 0.3, 0.5, 0.7, and 1 in our experiments. Fig. 10(b) illustrates the convergence tendencies of Cepe-FL with various values of μ. It is evident that the lack of balance does not have a substantial impact on the convergence of the model. We guess that this phenomenon might be attributed to the fact that local models can effectively acquire knowledge from IID data, even in cases where the data distribution among workers is uneven.

F. Influence of Batch Size b

In this subsection, we investigate how the size of the batch impacts the performance of Cepe-FL. Experiments are carried out on the task of Lenet5 on MNIST using IID data, with the parameters $\lambda = 0.1$ and $\mu = 1$. Fig. 10(c) depicts the convergence patterns of Cepe-FL with multiple batch sizes (32, 64, 100, and 128), respectively. In general, it can be stated that appropriately

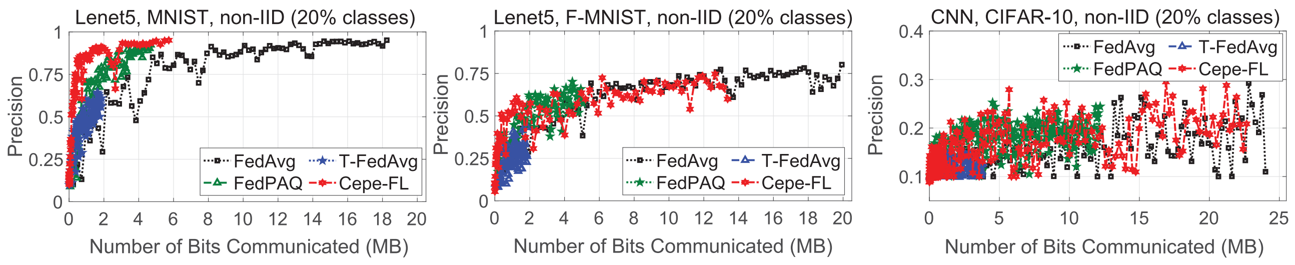

Fig. 9. Convergence trends and communication overheads with non-IID data (20% classes).

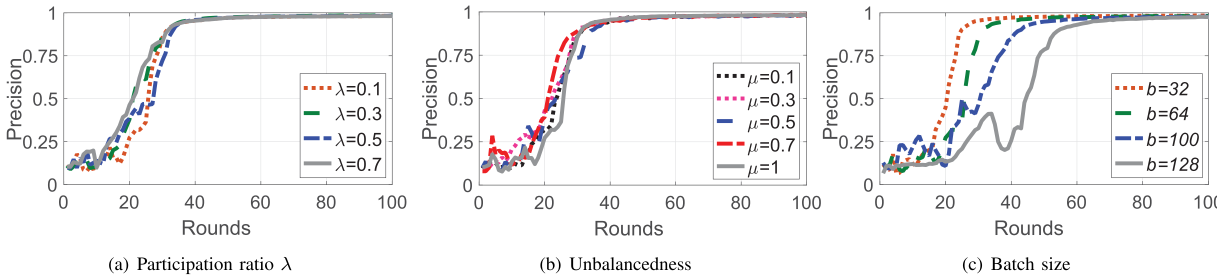

Fig. 10. Convergence curves of Cepe-FL on the task of Lenet5 on MINST (IID) in different FL environments.

raising the batch size has the potential to expedite and enhance the convergence of the model. It is noteworthy that an increase in batch size leads to a deceleration in the convergence of the model, accompanied by significant oscillations. During the training, we examined the fluctuation of compression ratios and discovered that a larger batch size results in a compression ratio that decreases at a slower rate. We guess that this leads to comparatively small reconstruction precision during the initial phase of federated training, thereby resulting in a slower and less stable convergence of the model.

G. Influence of Sparse Dictionary Learning Inputs

In this subsection, we investigate the influence of dictionary learning inputs on the performance of Cepe-FL. The MNIST dataset is utilized in this study to train the Lenet5 model for one thousand iterations. Quasi-validation sets of varying sizes, specifically 100 and 1000 samples, are generated. The parameter ζ is set to values of 100 and 200. Additionally, in order to pick the ζ intermediate models from the 1000 different models, we use three different methods: selecting the first and last ζ models, as well as selecting ζ models in an even manner. The convergence patterns of Cepe-FL under various configurations are illustrated in Fig. 11. It is observed that when dealing with a substantial quasi-validation set consisting of 1000 samples, the Cepe-FL demonstrates greater resilience, as seen by the fact that variations in both ζ and the selection approach have minimal impact on its convergence. In the case of the quasi-validation set consisting of 100 samples, it is evident that a higher value of ζ results in a more rapid convergence. Furthermore, the optimal selection approach involves choosing the most recent ζ models. Significantly, the experimental outcomes underline that Cepe-FL exhibits commendable performance in the fields of both test precision and convergence rate, even when utilizing a limited quasi-validation set.

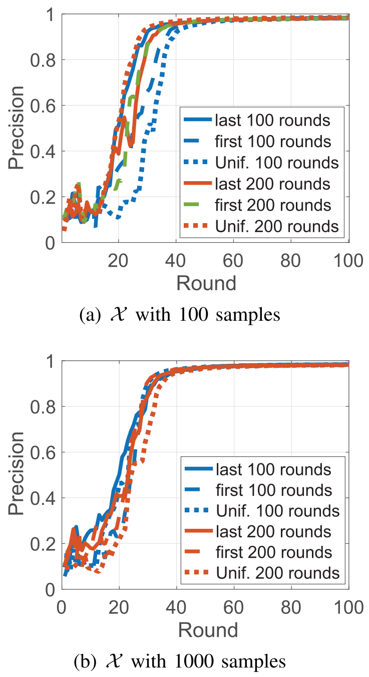

Fig. 11. Convergence curves of Cepe-FL on different input of dictionary learning.

H. Influence of the Quasi-Validation Set Distribution

In this experiment, we study how the performance of the FL model is affected when the distribution of quasi-validation set cannot fully represent the entire data distribution. To this end,

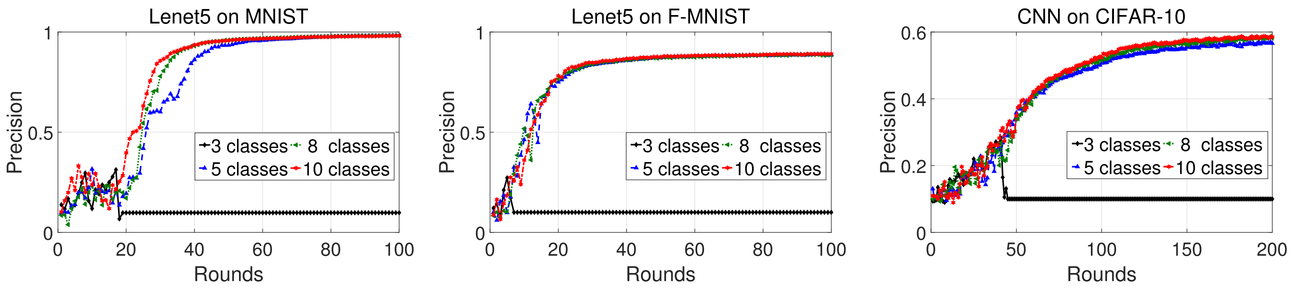

Fig. 12. Convergence trends of quasi-validation sets with different distributions.

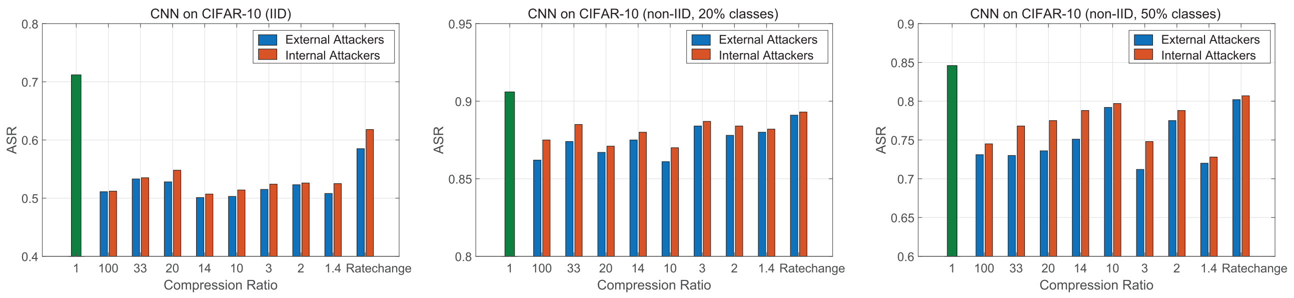

Fig. 13. ASR with different compression ratios on CNN on CIFAR-10.

for each FL task (containing 10 classes of images), we remove 7, 5 and 2 (i.e., left 3, 5, 8) classes of data samples in the corresponding quasi-validation set, respectively, without changing other settings. Experimental results are shown in Fig. 12. It can be seen, in all tasks, as long as the classes of samples are not excessively deleted, neither the convergence speed nor the final model accuracy will be significantly reduced. In the F-MNIST task, removing even half classes of samples almost not degrades FL performance in terms of convergence speed and final model accuracy. In the MNIST task, removing 2 or 5 classes of samples results in slightly slower convergence however does not affects the final model accuracy. In the CIFAR-10 task, the more classes of samples are removed, the lower the final model accuracy is. The final model accuracies are 58.7%, 57.9% and 56.4% when the numbers of classes are 10, 8 and 5, respectively. However, in all tasks, when 7 classes of samples are removed, the federated training fails due to large reconstruction deviation.

I. *Defending Against Membership Inference Attacks*

1) Implementation of Membership Inference Attacks: We focus on white-box membership inference attacks. White-box attackers know the compressed model parameters as well as hyperparameters. They may either be internal entities which take part in the federated training (i.e., participants or the server) or external attackers (i.e., outsiders). Specifically, internal attackers possess the knowledge of the sparse dictionary learnt by running K-SVD, and thus can reconstruct model parameters of victims. External attackers know nothing about the K-SVD-based sparse dictionary, they use DCT to reconstruct the model parameters. Furthermore, we consider that attackers are honest-but-curious, which means they will strictly follow the protocols of FL.

In our attack, the attacker first reconstructs the local model parameters of its victim, then it follows the membership inference attacks proposed by M. Nasr et al. [35], where the reconstructed parameters are used to train an inference model. The settings of the inference model are consistent with [35]. The batch size is 64, the learning rate is 0.0001, and the optimizer is Adam. Initializing the model weights requires a mean of 0, a variance of 0.01, and all bias values are set to 0.

Definition 3. (Attack Success Rate): A commonly used indicator to measure membership inference attacks, defined as

$$ASR = \frac{TP}{TP + FP} \tag{11}$$

where TP represents the number of samples correctly predicted as members of a training dataset and FP represents the number of non-member samples incorrectly predicted as members. A high attack success rate (ASR) indicates good attack performance. In our attacks, the attacker first randomly selects a participant as a victim, then trains the inference model by using the compressed local updates of the victim. To test the performance of the inference model, the attacker composes the testing (inferring) dataset in which half samples are members. Specifically, each inferring dataset contains 300 samples. 50% samples are randomly picked from the local dataset of the victim and 50% samples are randomly selected from the training samples of other participants.

2) Impact of Compression Ratio: This experiment is conducted on the CNN training task (with CIFAR-10). The attacker observes the local model parameters of its victim participant

TABLE VI
IMPACT OF OBSERVED ITERATIONS TO ASRS

Tasks	Observed Iterations	Compression Approach				No Compression
		10		Adaptive		
		External	Internal	External	Internal	
Lenet5 on MNIST	First 10 Rounds	0.500	0.509	0.515	0.537	0.572
	Last 10 Rounds	0.508	0.514	0.517	0.545	0.604
Lenet5 on F-MNIST	First 10 Rounds	0.502	0.521	0.508	0.527	0.599
	Last 10 Rounds	0.520	0.530	0.531	0.546	0.605
CNN on CIFAR-10	First 10 Rounds	0.487	0.506	0.501	0.514	0.681
	Last 10 Rounds	0.503	0.514	0.585	0.618	0.712

TABLE VII
IMPACT OF THE NUMBER OF OBSERVED MODELS TO ASRS

Tasks	The Number of Observed Models	Compression Approach				No Compression
		10		Adaptive		
		External	Internal	External	Internal	
Lenet5 on MNIST	5 Rounds	0.500	0.506	0.521	0.546	0.600
	10 Rounds	0.508	0.514	0.517	0.545	0.604
	15 Rounds	0.504	0.515	0.510	0.546	0.599
	20 Rounds	0.500	0.518	0.525	0.547	0.651
Lenet5 on F-MNIST	5 Rounds	0.513	0.518	0.522	0.545	0.606
	10 Rounds	0.520	0.530	0.531	0.546	0.605
	15 Rounds	0.516	0.522	0.522	0.546	0.613
	20 Rounds	0.520	0.528	0.525	0.541	0.602
CNN on CIFAR-10	5 Rounds	0.506	0.510	0.617	0.618	0.700
	10 Rounds	0.503	0.514	0.585	0.618	0.712
	15 Rounds	0.513	0.517	0.602	0.606	0.715
	20 Rounds	0.515	0.521	0.593	0.604	0.716

for 10 rounds, and trains the inference model with the observed model parameters for 100 rounds. We examine the ASRs of both internal and external attackers, under different compression ratios ($1\times$ refers to no compression) and adaptive compression ratios. We also verify the ASRs under different data distributions.

According to the experimental results shown in Fig. 13, we can make the following three conclusions. First, Cepe-FL can effectively suppress the ASRs, even under a small compression ratio like $1.4\times$. It can be seen that the ASRs are decreased up to 20%. This is attributed to the fact that deviations are inevitably introduced in the reconstructed model parameters, which reduces the accuracy of the inference model trained on them. The ASRs under adaptive compression ratio increase slightly. The reason is that small compression ratios close to 1 may be applied to accelerate model convergence, which leads to overfitting, and facilities inference attacks. Second, the ASRs of the external attacker are slightly lower than those of the internal attacker. This is because that the external attacker reconstructs model parameters using DCT as sparse basis, while the internal attacker utilizes K-SVD-based sparse dictionary which results in more accurate model reconstruction. Last, we observe that the ASRs to non-IID FL are notably higher than that to IID FL. Moreover, among the non-IID cases, large differences between data distributions on participants imply high ASRs. Under the setting that each participant only holds 20% class of samples in the global dataset, the ASRs are between 0.85 and 0.9. When the proportion of class held by each participant increases to 50%, the ASRs are typically below 0.8.

3) Impact of Observed Iterations: In this experiment, we vary the observed iterations of the victim's local models. The attackers observe local models in the first and last 10 rounds of FL, respectively. We conduct experiments on all three federated learning tasks on IID data. Both internal and external attackers launch inference attacks against federated training procedures with fixed ($10\times$) and adaptive compression ratio, and without compression.

From Table VI, we find that, for both types of attackers and under all settings, the ASRs relevant to the last 10 rounds are higher than those relevant to the first 10 rounds. That is because in the last 10 rounds, the models remember more information about the training data than in the first 10 rounds.

4) Impact of the Numbers of Observed Models: In this experiment, we examine the effect of the number of observed models t on ASR. The FL tasks and their settings are the same as that of the previous experiment. The attacker trains an inference model using its victim's models of the last t rounds as. We increase the value of t from 5 to 20 with a step of 5. From Table VII, we find that the ASR varies slightly (showing small fluctuation) under different values of t in all cases. We speculate that is because the observed models are close to convergence. Thus, increasing t does not improve the ASR significantly. For example, in the task trained with MNIST, when the values of t are 5 and 20, the ASR is both 0.5, the same as that for a random guess.

VIII. Conclusion

In this paper, we propose Cepe-FL, a compressed FL approach, which not only significantly improves the communication efficiency between the server and participants, but also suppresses membership inference attacks, providing stronger privacy protection. Cepe-FL applies compression to model parameters rather than to gradients and supports two-way compression. It utilizes a quasi-validation set held by the server to learn the sparse basis instead of using commonly used basis so as to maximize reconstruction errors. To achieve the best trade-off between the communication cost and the model performance, Cepe-FL adjusts the compression ratio adaptively. Moreover, by taking advantage of joint model reconstruction and layer-wise compression, the computational overhead on the server side can be dramatically reduced. We conduct experiments on three image classification tasks. Experimental results show that Cepe-FL demonstrates better performance than three SOTA comparisons in all tasks. We implement membership inference attacks launched by both internal and external white-box attackers. Experimental results indicate that the ASR drops on Cepe-FL, compared with uncompressed FL, which suggests that Cepe-FL enhances the local data privacy of participants.

References

[1] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, "SignSGD: Compressed optimisation for non-convex problems," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 560–569.

[2] W. Wen et al., "TernGrad: Ternary gradients to reduce communication in distributed deep learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1508–1518.

[3] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1707–1718.

[4] N. Strom, "Scalable distributed DNN training using commodity GPU cloud computing," in *Proc. 16th Annu. Conf. Int. Speech Commun. Assoc.*, Dresden, Germany, 2015, pp. 1488–1492.

[5] A. F. Aji and K. Heafield, "Sparse communication for distributed gradient descent," 2017, *arXiv: 1704.05021*.

[6] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," 2017, *arXiv: 1712.01887*.

[7] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-IID data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 9, pp. 3400–3413, Sep. 2019.

[8] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Sparse binary compression: Towards distributed deep learning with minimal communication," in *Proc. Int. Joint Conf. Neural Netw.*, 2019, pp. 1–8.

[9] J. Konečny, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016, *arXiv:1610.05492*.

[10] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[11] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "FedPAQ: A communication-efficient federated learning method with periodic averaging and quantization," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2020, pp. 2021–2031.

[12] F. Haddadpour, M. M. Kamani, A. Mokhtari, and M. Mahdavi, "Federated learning with compression: Unified analysis and sharp guarantees," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2021, pp. 2350–2358.

[13] J. Xu, W. Du, Y. Jin, W. He, and R. Cheng, "Ternary compression for communication-efficient federated learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 3, pp. 1162–1176, Mar. 2022.

[14] E. J. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Inf. Theory*, vol. 52, no. 2, pp. 489–509, Feb. 2006.

[15] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.

[16] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst.*, vol. 2, pp. 429–450, 2020.

[17] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 5132–5143.

[18] X. Yao, C. Huang, and L. Sun, "Two-stream federated learning: Reduce the communication costs," in *Proc. IEEE Vis. Commun. Image Process.*, 2018, pp. 1–4.

[19] T. Lin, S. U. Stich, K. K. Patel, and M. Jaggi, "Don't use large mini-batches, use local SGD," 2018, *arXiv: 1808.07217*.

[20] J. Wang and G. Joshi, "Adaptive communication strategies to achieve the best error-runtime trade-off in local-update SGD," in *Proc. Mach. Learn. Syst.*, 2019, pp. 212–229.

[21] Y. Xiong, R. Wang, M. Cheng, F. Yu, and C.-J. Hsieh, "FedDM: Iterative distribution matching for communication-efficient federated learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 16323–16332.

[22] D. Jhunjhunwala, A. Gadhikar, G. Joshi, and Y. C. Eldar, "Adaptive quantization of model updates for communication-efficient federated learning," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2021, pp. 3110–3114.

[23] M. M. Amiri, D. Gunduz, S. R. Kulkarni, and H. V. Poor, "Federated learning with quantized global model updates," 2020, *arXiv: 2006.10672*.

[24] S. Chen, C. Shen, L. Zhang, and Y. Tang, "Dynamic aggregation for heterogeneous quantization in federated learning," *IEEE Trans. Wireless Commun.*, vol. 20, no. 10, pp. 6804–6819, Oct. 2021.

[25] P. S. Bouzinis, P. D. Diamantoulakis, and G. K. Karagiannidis, "Wireless quantized federated learning: A joint computation and communication design," *IEEE Trans. Commun.*, vol. 71, no. 5, pp. 2756–2770, May 2023.

[26] A. Sahu, A. Dutta, A. M. Abdelmoniem, T. Banerjee, M. Canini, and P. Kalnis, "Rethinking gradient sparsification as total error minimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 8133–8146.

[27] P. Han, S. Wang, and K. K. Leung, "Adaptive gradient sparsification for efficient federated learning: An online learning approach," in *Proc. IEEE 40th Int. Conf. Distrib. Comput. Syst.*, 2020, pp. 300–310.

[28] M. K. Nori, S. Yun, and I.-M. Kim, "Fast federated learning by balancing communication trade-offs," *IEEE Trans. Commun.*, vol. 69, no. 8, pp. 5168–5182, Aug. 2021.

[29] A. M. Abdelmoniem and M. Canini, "DC2: Delay-aware compression control for distributed machine learning," in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.

[30] R. Jin, P. Dai, and K. Xiong, "Communication-efficient federated learning with channel-aware sparsification over wireless networks," in *Proc. 57th Annu. Conf. Inf. Sci. Syst.*, 2023, pp. 1–6.

[31] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 14774–14784.

[32] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients-how easy is it to break privacy in federated learning?," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 16937–16947.

[33] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the GAN: Information leakage from collaborative deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 603–618.

[34] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *Proc. IEEE Symp. Secur. Privacy*, 2019, pp. 691–706.

[35] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *Proc. IEEE Symp. Secur. Privacy*, 2019, pp. 739–753.

[36] X. He, Y. Xu, S. Zhang, W. Xu, and J. Yan, “Enhance membership inference attacks in federated learning,” *Comput. Secur.*, vol. 136, 2023, Art. no. 103535.
[37] T. Nguyen, P. Lai, K. Tran, N. Phan, and M. T. Thai, “Active membership inference attack under local differential privacy in federated learning,” 2023, *arXiv:2302.12685*.
[38] Y. Gu, Y. Bai, and S. Xu, “CS-MIA: Membership inference attack based on prediction confidence series in federated learning,” *J. Inf. Secur. Appl.*, vol. 67, 2022, Art. no. 103201.
[39] M. A. Rahman, T. Rahman, R. Laganière, N. Mohammed, and Y. Wang, “Membership inference attack against differentially private deep learning model,” *Trans. Data Privacy*, vol. 11, no. 1, pp. 61–79, 2018.
[40] J. Chen, W. H. Wang, and X. Shi, “Differential privacy protection against membership inference attack on machine learning for genomic data,” in *Proc. Proc. Pacific Symp.*, 2020, pp. 26–37.
[41] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, “Membership inference attacks against machine learning models,” in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 3–18.
[42] J. Jia, A. Salem, M. Backes, Y. Zhang, and N. Z. Gong, “MemGuard: Defending against black-box membership inference attacks via adversarial examples,” in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2019, pp. 259–274.
[43] M. Nasr, R. Shokri, and A. Houmansadr, “Machine learning with membership privacy using adversarial regularization,” in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 634–646.
[44] E. Candes and J. Romberg, “Sparsity and incoherence in compressive sampling,” *Inverse Problems*, vol. 23, no. 3, 2007, Art. no. 969.
[45] Y. Tsaig and D. L. Donoho, “Extensions of compressed sensing,” *Signal Process.*, vol. 86, no. 3, pp. 549–571, 2006.
[46] E. J. Candes, “The restricted isometry property and its implications for compressed sensing,” *Comptes Rendus Mathematique*, vol. 346, no. 9/10, pp. 589–592, 2008.
[47] L. Deng, “The MNIST database of handwritten digit images for machine learning research [best of the web],” *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, Nov. 2012.
[48] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms,” 2017, *arXiv: 1708.07747*.
[49] A. Krizhevsky and H. Geoffrey, “Learning multiple layers of features from tiny images,” 2009.
[50] J. A. Tropp and A. C. Gilbert, “Signal recovery from random measurements via orthogonal matching pursuit,” *IEEE Trans. Inf. Theory*, vol. 53, no. 12, pp. 4655–4666, Dec. 2007.
[51] M. Aharon, M. Elad, and A. Bruckstein, “K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation,” *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.
[52] K. Das, J. Jiang, and J. Rao, “Mean squared error of empirical predictor,” *Ann. Statist.*, vol. 32, no. 2, pp. 818–840, 2004.

Ye Liu (Member, IEEE) received the MS degree from Department of software engineering, Kunming University of Science and Technology, in 2019. She is currently working toward the PhD degree from the School of Computer Science and Technology, Donghua University. Her research interests include ubiquitous and pervasive computing, privacy and security of federated learning.

Shan Chang (Member, IEEE) received the PhD degree in computer software and theory from Xi’an Jiaotong University, Xi’an, China, in 2012. From 2009 to 2010, she was a visiting scholar with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong. She was also a visiting scholar with BBCR Research Lab, University of Waterloo, Waterloo, ON, Canada, from 2010 to 2011. She is currently a professor with the Department of Computer Science and Technology, Donghua University, Shanghai, China. Her research interests include security and privacy in mobile networks and sensor networks. She is a member of Communication Society, and Vehicular Technology Society.

Yiqi Liu received the undergraduation degree from the Department of Computer Science and Technology, Donghua University, in 2023. He is currently working toward the master’s degree with the Institute of Computing Technology, Chinese Academy of Sciences. His research interests include federated learning and incentive design.