2022 IEEE 28th International Conference on Parallel and Distributed Systems (ICPADS) | 978-1-6654-7315-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICPADS56603.2022.00011

FedCS: Communication-Efficient Federated Learning with Compressive Sensing

Ye Liu
Computer Science and Technology
Donghua University
Shanghai, China
liuye@mail.dhu.edu.cn

Shan Chang*
Computer Science and Technology
Donghua University
Shanghai, China
changshan@dhu.edu.cn

Yiqi Liu
Computer Science and Technology
Donghua University
Shanghai, China
liuyiqi@mail.dhu.edu.cn

***Abstract*—In Federated Learning (FL), two-way model exchanges are required between the server and the workers every training round. Due to the large size of machine learning models, communications between them lead to high training delay and economic cost. At present, communication-efficient FL methods, for examples, top-k sparsification and quantization, taking advantages of the sparseness of model gradients and the fact that gradient-based model updating can tolerance small deviations, effectively reduce the communication cost of single training round. However, these gradient-based communication-efficient schemes cannot be applied to downlink communication. In addition, they cannot be used in conjunction with those communication-frequency-suppressed methods, e.g., FedAvg, which hinders them from further improving training efficiency. In this paper, we propose FedCS, a compressive sensing based FL method, which can effectively compress and accurately reconstruct non-sparse model (both local and global) parameters (weights), and can reduce the overall communication cost up to 10× as compared to FedAvg without decreasing test accuracy. We introduce 1) a dictionary learning scheme with a quasi-validation set, which helps to project non-sparse parameters onto a sparse domain; 2) a joint reconstruction scheme, by using which the server recovers global model parameters by executing the reconstruction algorithm only once a round, regardless of the number of compressed local models; 3) a compression ratio adjustment strategy, which balances the trade-off between total communication cost and model accuracy. We perform FedCS on three image classification tasks, and compare it with FedAvg, FedPAQ and T-FedAvg (two improvements of FedAvg). Experimental results demonstrate that FedCS outperforms comparison methods in all tasks, and always maintains a comparable test accuracy to FedAvg, even using a small quasi-validation set and on Non-IId data.**

***Index Terms*—federated learning, communication-efficient, compressive sensing, dictionary learning**

I. Introduction

Federated learning (FL) is a distributed machine learning architecture that allows multiple participants to jointly train a global model without sharing their private data. In FL, the training (update) of the global model requires several rounds. In each round, a central server randomly selects some clients to participate in training. Based on a current global model, the selected workers use their local data to update the global model, and upload their local updates of models to the server for aggregation, obtaining an updated global model. The advantages of FL are as follows: first of all, since clients do not need to publish their training data to any third party (they only upload model updates), FL naturally has the ability (feature) of protecting private data of clients. Second, the training data are always stored on the clients, which saves the storage costs on the server side. Third, intensive computations required to train the complex models are distributed to a large number of clients, and the server only needs to run the aggregation protocol, which greatly reduces the computational overhead on the server side. However, distributed training also raises a series of problems. One of the main problems is the communication cost and training time delay caused by the transmission of model updates. Specifically, for modern architectures with millions of parameters, the size of which may be gigabytes, during hundreds thousands of training iterations on large datasets, the total communication per client can easily exceed PB. Additionally, unreliable network conditions of clients can cause severe delays. In other words, under limited bandwidth and unstable network transmission conditions, massive communications significantly degrade the performance of FL. Therefore, finding a communication-efficient FL method is crucial.

In general, the reduction of communication cost can be done in two ways. First, reduce the amount of communication load per round. There has been some valuable work based on this idea. For example, literatures [1] [2] [3] take advantage of the fact that the gradient represents the direction on which the model loss is minimized, thus has a certain ability to tolerate deviations, and reduce the amount of communication through gradient quantization. However, the compression methods based on gradient quantization can only obtain a relatively weak compression ratio. By applying gradient quantization-based compression methods, it is difficult to achieve a compromise between fast model convergence speed and high compression ratio. Literatures [4] [5] [6] take advantage of the sparseness of gradients, and only upload the top-k gradients, and thus effectively reduce the single-round traffic volume. Literature [5] shows that even if only 0.1% gradients are retained, the model can still quickly converge to high precision. Literatures [7] [8] [9] combine quantization with sparsification to compress model updates, reducing communication overhead further. The common problem of these methods is performing compression on model gradients,

* is the corresponding author.

978-1-6654-7315-6/23/$31.00 ©2023 IEEE
DOI 10.1109/ICPADS56603.2022.00011

which implies the disadvantage that they cannot be used for downlink communication, i.e., communication from server to clients. This is because that, usually, different clients will be selected as workers in different training rounds, which means the versions of the global model held by the dynamically joined clients in each round are different. To keep all clients synchronized, the server broadcasts the latest global model instead of the incremental update, i.e.,model gradients, to all workers during downlink communication. Second, reduce the total number of communication rounds. In other words, reduce the total numbers of communication rounds between the workers and the server during training. A classic work is the FedAvg algorithm proposed by literature [10], which reduces the number of interactions between the clients and the server by increasing the number of local iterations. In FedAvg, workers no longer upload gradients for each batch, and more computation is added to each worker by iterating the local update multiple times before the averaging step. If the above two types of compression methods can be combined at the same time, a higher compression rate will be obtained [11] [12] [13]. Thus, we can conclude that only conducting compression on model parameters rather than gradients has the potential to satisfy the following three conditions at the same time: reducing the total number of communications, communication cost of each round, and supporting downlink compression. Literature [13] has carried out related studies, but the disadvantage is that the computational cost is too high.

In this paper, we aim to design a communication-efficient FL method that supports bidirectional communication compression, and further reduces the communication cost of each single round on the basis of FedAvg, to ultimately improve the performance of FL training. To this end, we propose FedCS, a compressive sensing strategy to compress model parameters rather than gradients. FedCS does not rely on the sparsity of model parameters and can support both upstream and downstream compressions, resulting a large reduction of the total amount of communication cost during federated training. According to compressive sensing, a sparse signal can be recovered accurately from highly incomplete measurements, i.e., the number of samples can be far less than that required by the Shannon-Nyquist sampling theorem [14] [15]. In FL, to minimizing the impact of compression on model convergence, it is desired to compress the parameters of a model with large compression ratio, and then reconstruct them with high accuracy, which is however very challenging. First, model parameters are usually non-sparse, and it is difficult to find the sparse representations of them. Commonly used way to generate sparse dictionaries, such as Discrete Cosine Transform(DCT) or Discrete Fourier Transforms(DFT), have proven to be ineffective in projecting model parameters to their sparse representations. Second, reconstruction algorithms are usually computation-intensive. In a large scale FL system, reconstructing compressed local models one by one brings about too much computational cost on the server side, which severely delays model updating. Third, a high compression ratio naturally corresponds to a relatively low reconstruction accuracy, using an excessively high compression ratio may cause a large reconstruction error, which results in a decrease of the global model accuracy, and slow model convergence. Therefore, it is difficult and important to determine the best compression ratio which not only reduces the amount of communication as much as possible, but also guarantees both high test accuracy and fast model convergence.

To address the above challenges, we first perform dictionary learning with a quasi-validation dataset held by the server to learn the sparse representation of model parameters. Next, on the server side, taking advantage of the linearity of compression, we reduce the computational cost of global model recovery from n (the number of workers) to 1 execution of the reconstruction algorithm. Moreover, we reduce the computational cost of compression through layer-wise compression. Finally, we propose an adaptive compression ratio selection algorithm, which determines the appropriate compression ratio according to the model training loss, so as to maximize the communication compression ratio and minimize the model loss. We empirically evaluate FedCS on three image classification tasks and compare its performance with FedAvg [10], FedPAQ [11] and T-FedAvg [13] on both IID and Non-IID data. We also examine how different federation learning environments impact the performance of FedCS. The experimental results suggest that FedCS outperforms the other three comparison schemes in all cases, and even using a small quasi-validation set to learn the sparse dictionary can achieve high reconstruction accuracies.

II. Related Work

Several methods have been proposed to reduce the amount of communication during the training process. In general, there are three kinds of strategies: postponing communication, sparsification, and quantization.

Postponing Communication: refers to reducing the communication frequency. Mcmahan *et al.* [10] proposed the FedAvg algorithm, which combines local SGD with the server that performs model averaging. The client first iterates local updates multiple times and then sends the local iteration results to the server, reducing the number of necessary communication cycles ×10 to ×100, but FedAvg has poor performance when the data is non-IID. Sahu *et al.* [16] proposed a more general FedProx algorithm, which can dynamically update the number of local calculations required by different clients in each round, without requiring the participants to unify the number of operations in each update. Therefore, FedProx optimizes more effectively when the data is non-IID. The federated averaging method reduces upstream and downstream communication and is suitable for large numbers of clients and partial client participation.

Sparsification: refers to clients uploading only a small portion of data with significant characteristics. Strom *et al.* [4] only transmit gradients larger than a certain threshold. In order to prevent information loss, the remaining "unimportant" gradients are accumulated locally, and are transmitted when they are accumulated to the threshold, this method can achieve

compression ratios up to 3 orders of magnitude. However, in practice, it is difficult to choose an appropriate value for the threshold, so Aji & Heafield *et al.* [5] proposed a method to sparse gradients by the sparsity rate, that is, truncate the smallest gradient smaller than the sparsity rate, and only transmit the remaining large gradients to the server, other gradients are also accumulated in a residual, which is accumulated and transmitted. When the sparsity rate of this method is 0.1%, the convergence performance and accuracy of the model decrease slightly. Lin *et al.* [6] combined the sparse gradient with momentum correction, momentum factor masking, and training warm-up to further improve the compression effect, this method achieves compression ratios as high as ×270 to ×600 on different models.

Quantization: refers to clients quantizing the transmitted value from a 32-bit floating point number to some lower bit-width representation. Bernstein *et al.* [1] proposed signSGD to quantize gradients into binary symbols, achieving a compression ratio of ×32. Wen *et al.* [2] proposed TernGrad, which stochastically quantized the gradient into ternary values $\{0, 1, -1\}$, and achieved a compression ratio of ×16. Alistah *et al.* [3] proposed QSGD, which is a convergence-guaranteed compression scheme, quantizing 32-bit floating point numbers into 8-bit integers, achieving a compression ratio of ×4.

III. System Model

We focus on a federated learning architecture, which consists of a central server $\mathcal{S}$, and N distributed participants $p_i, i \in \{1, ..., N\}$. Each participant p_i possesses a private dataset $\mathcal{D}_i = \{x_{i,d}, y_{i,d}\}, d \in \{1, ..., |\mathcal{D}_i|\}$, where $x_{i,d}$ and $y_{i,d}$ represents each data sample and its corresponding label. The goal of the system is to learn a global model which can generalize well on testing input after aggregating over the training results, i.e., local models, from participants.

Particularly, we consider the federated system follows the *Fedavg* learning method. The learning process is composed of several synchronous rounds. During round t, the server selects l ($l \leqslant N$, and $\lambda = l/N$ is participation ratio) participants p_i^t as workers, each of which downloads the latest shared model $\mathbb{G}^{t-1}$ from the server, and individually processes $\mathbb{G}^{t-1}$ by running Mini-Batch Gradient Descent (MBGD), with its private dataset, for several times, obtaining a local model $\mathbb{L}_i^t$, and uploads $\mathbb{L}_i^t$ to the server for aggregating. The central server updates the global model $\mathbb{G}^t$ by applying a weighted-averaging aggregation rule

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

where $\mathbf{D}^t$ refers to the total number of training examples in round t, i.e., $\mathbf{D}^t = \sum_{i=1}^{l} |\mathcal{D}_i|$. These steps are repeated in multiple rounds until the learning process converges.

Moreover, we introduce the concept of a *quasi-validation set*, which implies a collection of data samples following a similar (but not necessarily identical) distribution as the true distribution of all samples (across participants), i.e., $\{\mathcal{D}_1, .., \mathcal{D}_N\}$, in the FL system. In practice, the server can randomly collect a small number of data samples from similar data domains, and label them manually, to form the quasi-validation set.

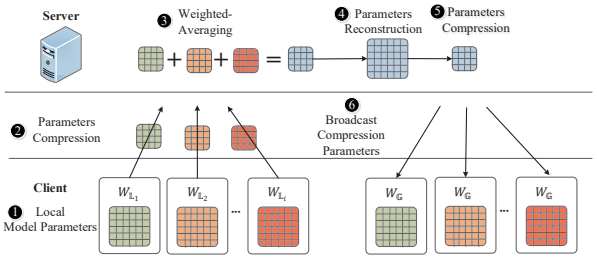


Fig. 1: The procedures performed by the server and workers to update the global model.

IV. The Design of FedCS

A. Preliminaries

FedCS enables communication-efficient federated learning by integrating *Fedavg* with *compressive sensing*. Thus, before explaining our FedCS design, we briefly introduce the theory of compressive sensing first.

Given a real-valued, finite-length, one-dimensional, discrete-time signal $\mathcal{W}$, which can be viewed as an $n \times 1$ column vector in $\mathbf{R}^n$, to be sensed, if $\mathcal{W}$ is sparse or nearly sparse (in original or some transform domain), we can obtain more directly a compressed version of $\mathcal{W}$ by taking only a small amount of linear and nonadaptive measurements, and (accurately or approximately) reconstruct $\mathcal{W}$ from that vastly compressed version $\mathcal{Y} \in \mathbf{R}^m$ (m can be far less than n) using efficient recovery algorithms [14] [15]. Mathematically, the sensing process of $\mathcal{W}$ can be represented as

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

where Φ is m-by-n *measurement matrix*, and should satisfy RIP (Restricted Isometry Property) [17].

Definition (Data Compression Ratio) is defined as the ratio between the uncompressed size and compressed size:

$$CompressionRatio(\delta) = \frac{UncompressedSize}{CompressedSize}$$

After applying compressive sensing, signal $\mathcal{W}$ is reduced to $\mathcal{Y}$, the compression ratio of $\mathcal{W}$ is n/m.

B. Overview

In our FedCS, before federated training, there exists an initialization step, which includes:

On the server side, it selects and initializes a (global) machine learning model $\mathbb{G}^0$ to be trained, whose parameters, denoted by $\mathcal{W}_{\mathbb{G}^0}$, are n-dimensional, and prepares an $n \times n$ Gaussian random matrix as the measurement matrix Φ, and learns an $n \times k$ ($k > n$) sparse dictionary Ψ by using the quasi-validation set $\mathcal{X}$. Ψ will be used to project model parameters communicated between server and workers, to some sparse

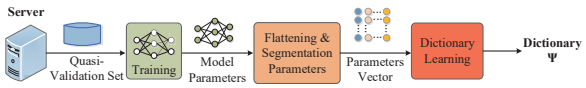


Fig. 2: The server learns a sparse dictionary by using quasi-validation set.

representations, since original $\mathcal{W}_{\mathbb{G}^0}$ are not sparse (see in subsection IV-C).

On the worker side, they retrieve Φ from the server, which will be used to construct a run-time measurement matrix in each training iteration.

Then, the server and works follow the following procedures to obtain an updated model $\mathbb{G}^1$ (see Fig. 1):

First, the server decides compression ratio δ (see subsection IV-D), which implies the compression version of $\mathcal{W}_{\mathbb{G}^0}$ is m-dimensional ($m = n/\delta$), and publishes δ to all workers.

Second, after getting $\mathbb{G}^0$, each worker p_i runs MBGD iteratively using its local dataset $\mathcal{D}_i$ until obtaining the local model $\mathbb{L}_i^1$, and flattens $\mathcal{W}_{\mathbb{L}_i^1}$ into an $n \times 1$ column vector, and considers $\mathcal{W}_{\mathbb{L}_i^1}$ as the signal to be compressed. Then, m rows in Φ will be extracted (according to a public selection algorithm, for example selecting first m rows) to form a run-time Φ_δ used to compress $\mathcal{W}_{\mathbb{L}_i^0}$ according to Equation (2), i.e., $\mathcal{C}_{\mathbb{L}_i^1} = \Phi_\delta \mathcal{W}_{\mathbb{L}_i^1}$, and transmits $\mathcal{C}_{\mathbb{L}_i^1}$ to the server.

Third, after receiving $\mathcal{C}_{\mathbb{L}_i^1}$, the server runs decompression algorithm, such as OMP [18] , to recover $\mathcal{W}_{\mathbb{L}_i^1}$, and applies the weighted-averaging aggregation protocol, obtaining $\mathbb{G}^1$. However, recovering local models one by one on the server side induces too much computational cost, thus we propose a joint recovery scheme in which the server only needs to conduct reconstruction algorithm once in each iteration (see in subsection IV-E1).

Notice that it is easy to apply FedCS to downlink stream compression, in which the server compresses $\mathbb{G}^1$ by using Φ_δ, and the workers recover $\mathbb{G}^1$ using $\Phi\Psi$ which can be obtained from the server in the initialization step.

C. Learning Sparse Dictionary Ψ

According to the principle of compressed sensing [14] [15], the sparse representation of model parameters (i.e., weights) $\mathcal{W}_{\mathbb{L}_i^t}$, is the premise of recovering it accurately. Since $\mathcal{W}_{\mathbb{L}_i^t}$ themselves are not sparse, we should project them to a sparse domain. To this end, we need to exploit a sparse representation basis Ψ, named sparse dictionary, such that $\Psi \mathcal{S}_{\mathbb{L}_i^t} = \mathcal{W}_{\mathbb{L}_i^t}$, 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.

However, directly applying commonly used predefined sparse dictionaries, such as wavelet and localized Fourier basis, cannot obtain satisfactory sparsity on different models. Moreover, in federated training, characteristics of model parameters will change in different iterations, and it is difficult to find a dictionary that is effective for all models.

We utilize a dictionary learning method to find a sparse representation of $\mathcal{W}_{\mathbb{L}_i^t}$ (see Fig. 2). One of the key principles of dictionary learning is that the dictionary has to be inferred from the input data. However, the server has no knowledge about $\mathcal{W}_{\mathbb{L}_i^t}$ during learning. Thus, the server uses the small quasi-validation set $\mathcal{X}$, to train the global model $\mathbb{G}$ iteratively until convergence (or for a certain number of iterations), and evenly selected ζ intermediate models $\mathbb{G}^t$ obtained from each round of training. Then, to learn Ψ, the server uses parameters of ζ recorded intermediate models, i.e., $\mathcal{W}'_{\mathbb{G}^t}$, as input to run K-SVD algorithm, which works by iteratively alternating between the sparse representation of $\mathcal{W}'_{\mathbb{G}^t}$, i.e., $\mathcal{S}'_{\mathbb{G}^t}$ based on the current dictionary Ψ^k, and updating the dictionary to better fit the input [19].

The reason we use $\mathcal{W}'_{\mathbb{G}^t}$ to learn Ψ is because the data distribution of $\mathcal{X}$ is similar to the true sample distribution of workers. It means that the model trained by $\mathcal{X}$ is similar to those trained by samples across participants, i.e., $\{\mathcal{D}_1, .., \mathcal{D}_N\}$ (under the premise of the same model structure and parameters). Therefore, it is reasonable to learn Ψ by using a collection of $\mathcal{W}'_{\mathbb{G}^t}$.

D. Determining Compression Ratio δ

Generally, there is a trade-off between compression ratio, communication cost and model accuracy. The higher the compression ratio δ, the smaller the traffic volume, however the lower the reconstruction accuracy. We emphasize that, in FedCS, using a constant δ throughout FL does not achieve the best performance. This is because in the early stage of FL, the model parameters change greatly and thus could tolerate a relatively large reconstruction error. However, at the later stage of training, the model is close to convergence, and the model parameters are only fine-tuned. Large reconstruction error makes the accuracy of the reconstructed model unable to improve continuously. The convergence speed may drop significantly, which also implies more communication rounds, and thus more communication overhead. It even causes the drop of model accuracy and training failure. Therefore, it is necessary to specify an appropriate δ for each round of training, so as to minimize the total communication cost without incurring significant accuracy degradation.

We propose an adaptive compression ratio adjustment strategy. It can not only ensure that the final model accuracy is almost the same as the corresponding baseline model Fedavg, but also close to the minimum communication cost. The basic idea is to use a large δ at the beginning of training, and then gradually reduce δ according to the loss of the global model in subsequent training. The reason we adjust δ according to model loss is the decreasing trend of model loss is inversely related to the model convergence. The server adjusts δ using the following procedures.

- **Step 1:** Determine relevant parameters, including the maximum training rounds θ with constant compression ratio, initial compression ratio δ^0 which will be used in the previous θ training rounds, and ϵ, a scaling factor which is slightly smaller than 1.

- **Step 2:** Record the loss of the global model $\mathbb{G}^0$ after one round training, denoted by ℓ^0.
- **Step 3:** In round t $(t > \theta)$, the server calculates the average loss of the last θ rounds, denoted as ℓ^t, and calculates $\gamma = \ell^t/\ell^0$. Notice that ℓ^t decreases with t, and is non-negative, which implies $0 \leqslant \gamma \leqslant 1$. The server sets new compression ratio as $\delta^t = \gamma \cdot \delta^0$.
- **Step 4:** If the server observes that the model loss is unchanged for θ rounds, which implies either the recent models are converged, or have large reconstruction errors, then the server reduces δ^t to $\epsilon \cdot \delta^t$, attempting to further improve the model accuracy.

E. Reducing Computational Overhead

1) Jointly Recovering Model Parameters: Generally speaking, reconstruction algorithms are usually computation-intensive. For example, the computational complexity of OMP is $O(mns)$ [18] . Thus, in a training round with l workers, the computational costs to recover all local models are $l \cdot O(mns)$. In a large-scale FL system, is desired to reduce the computational complexity of reconstruction. Fortunately, we find that it is possible to reduce it from $l \cdot O(mns)$ to $O(mns)$. In other words, for each update of the global model, the server jointly reconstructs model parameters once no matter the number of local models.

Theorem (Linearity of Compression)**.** For any $m \times n$ measurement matrix Φ, two $n \times 1$ signal $\mathcal{W}_i$ and $\mathcal{W}_j$, and constant a_i and a_j,

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

The equation suggests that the server can perform weighted-averaging before recovering a global model. According to Equation 1, the following equation holds:

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

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

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

Consequently, the we can perform local model aggregation first, then reconstruct $\mathcal{W}_{\mathbb{G}^t}$ using the aggregated (compressed) local models.

Furthermore, joint reconstruction has two advantages. First, reconstruction error will be induced only once in each update. Second, due to the global model is trained by multiple workers, the distribution of training samples is more likely the same as quasi-validation set, low reconstruction error can be guaranteed, even workers hold Non-IID data.

2) Layer-wise Compression: Consider that workers are usually low-end mobile devices, and the number of model parameters can be very large. We use layer-wise compression to reduce compression cost. For example, a κ-layer model whose parameters are n-dimensional. Without losing generality, we assume that each layer is of equal size, and compression is $\delta = m/n$, which means the measurement matrix Φ is $m \times n$. Then computational complexity of compressing the model as a whole is $O(n \cdot m)$. By contrast, if compressing layer by layer, the size of Φ changes to $\frac{m}{\kappa} \times \frac{n}{\kappa}$, then the compression cost can be reduced to $\kappa \cdot O(\frac{m \cdot n}{\kappa^2})$ (i.e., $O(\frac{m \cdot n}{\kappa})$). Thus, we conclude that layer-wise compression makes the computational cost reduced to $1/\kappa$ of its original cost.

V. Experiment

We empirically validate our FedCS on three different learning tasks and compare its performance with FedAvg [10], FedPAQ [11] and T-FedAvg [13] in terms of model accuracy and communication cost, and examine how different federation learning environments impact its performance.

A. Methodology

Models and Data Sets: We implement our algorithm on PyTorch's distributed library to simulate real-world learning scenarios in FL. In this experiment, we use MNIST [20] and Fashion MNIST [21] to train a seven-layer Lenet5 model, and CIFAR10 [22] to train a six-layer CNN. Table I summarizes the hyper-parameters and optimizer used in each learning task.

TABLE I: Optimizer and Hyper-parameters

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

Compared Methods: We compare our FedCS with the following methods:

- *FedAvg:* classical communication efficient federated averaging method [10].
- *FedPAQ:* combines federation averaging and quantization methods [11] , quantizing 32-bit to 8-bit and achieving a compression ratio of ×4 in each training round;
- *T-FedAvg:* quantitative federated learning approach [13], which quantizes 32-bit to 2-bit, achieving a compression ratio of ×16 in each training round.

Additionally, in our FedCS, we do not compress the first convolutional layers of Lenet5 and CNN models because they are relatively small, and the relevant parameters are summarized in Table II. In all methods, the total number of workers is 100.

TABLE II: Relevant Parameters in FedCS

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

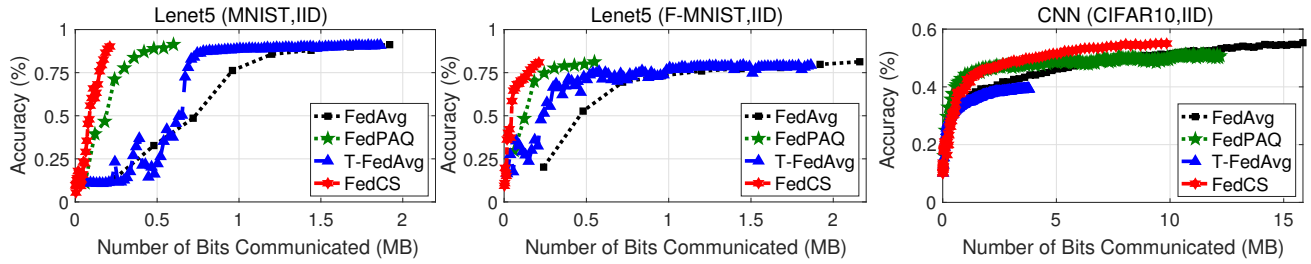


Fig. 3: Convergence trends and communication costs with IID data.

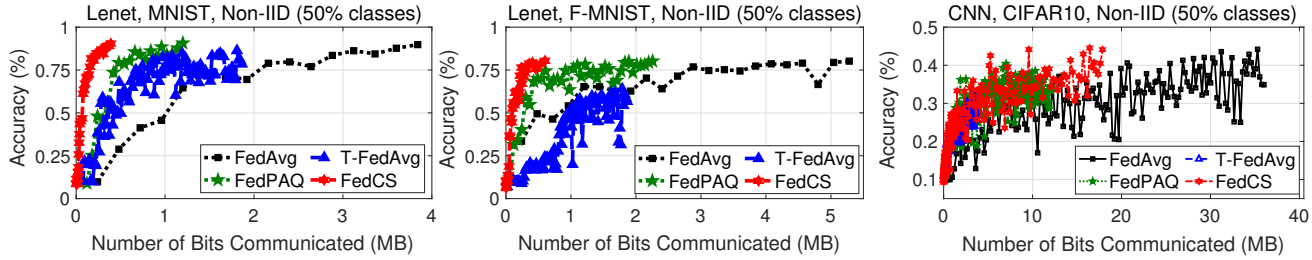


Fig. 4: Convergence trends and communication costs with non-IID data (50% classes).

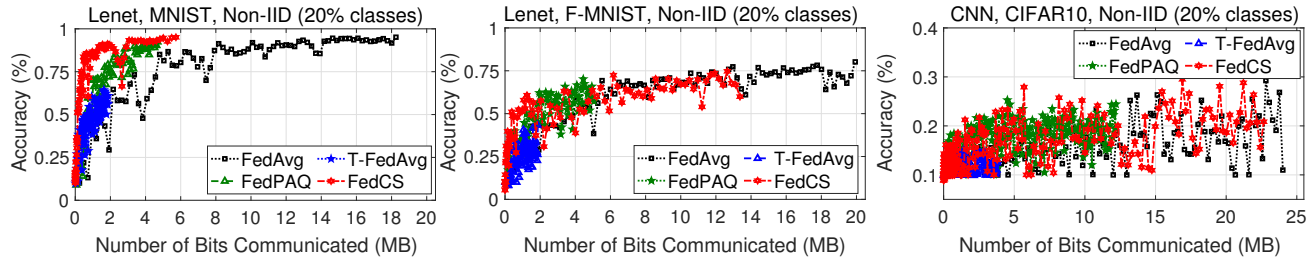


Fig. 5: Convergence trends and communication costs with non-IID data (20% classes).

B. Performance Comparison

In this experiment, the workers split all training data equally. The participation ratio of workers in each training round is 0.1. We conduct experimental comparing the communication costs and performance, i.e., testing accuracy, of four methods, with both IID (each worker holds all classes in the training data), Non-IID (i.e., each worker holds half and 20% classes in the training data) data. Table III shows the test accuracies, and upstream/downstream communication costs of different schemes after training 100 rounds (FedAvg and FedPAQ do not perform downlink compression). It can be seen that FedCS achieves comparable accuracy with FedAvg in all three tasks, however saved 76.2%, 39.7%, 71.6% communication costs, respectively. Table IV lists test accuracies of different schemes after training a certain numbers of rounds (Lenet5: 100 rounds, CNN: 200 rounds). Notice that data distributions do not affect the communication costs when the number of training rounds is fixed, thus we do not list them in the table. It can also be seen that test accuracies of FedCS are closest to that of FedAvg than other two schemes in all tasks.

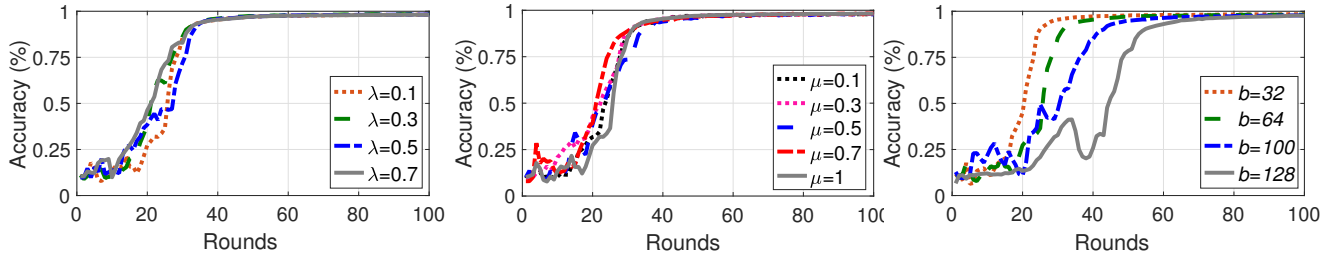
We plot the convergence curves on different data distributions in Fig. 3 (IID), Fig. 4 (Non-IID, 50% classes) and Fig. 5 (Non-IID, 20% classes), which illustrate the average number of bits needed to be uploaded by workers to achieve certain testing accuracy. From these results, we find that our FedCS outperforms the other three schemes in all cases. In the task of Lenet5 on MNIST, the final communication costs of FedCS are only around $1/10$, $1/3$ and $1/4$ of that of FedAvg, FedPAQ, and T-FedAvg, respectively, no matter what the data distribution is. In the task of Lenet5 on F-MNIST, we find that on Non-IID data, FedCS and FedAvg can still maintain the final test accuracies as on IID data, however that of FedPAQ and T-FedAvg suffer varying degrees of drops. The bigger the differences between data distributions of workers, the more the accuracy drops. In the task of CNN on CIFAR10, on Non-iid data, although accuracies of all schemes drop, FedCS always shows similar final accuracies with FedAvg, while final accuracies of the other two schemes are significantly

TABLE III: Test accuracy and communication cost of different algorithms when trained on IID data

Tasks	Lenet5 on MNIST		Lenet5 on Fashion-MNIST		CNN on CIFAR-10	
	Accuracy	Communication costs (KB) Upload/Download	Accuracy	Communication costs (KB) Upload/Download	Accuracy	Communication costs (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
FedCS	0.980	5712/5712	0.892	14472/14472	0.535	6828/6828

TABLE IV: Test accuracy of different algorithms 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
FedCS	0.956	0.975	0.788	0.868	0.287	0.445



(a) Participation ratio λ (b) Unbalancedness (c) Batch size

Fig. 6: Convergence curves of FedCS on the task of Lenet5 on MINST (IID) in different FL environments.

lower than that of FedAvg, for example, on IID data, the final accuracy of FedAvg is about 55%, while that of T-FedAvg is 40%. Furthermore, FedCS can save 35%-50% communication costs of FedAvg.

C. Influence of the Participation Ratio λ

We investigate the effect of λ on FedCS in this subsection. The experiments are done with IID data. Given a λ, we select $\lambda \cdot 100$ workers randomly in each training round. Fig. 6(a) depicts the test accuracies achieved by FedCS during training with different participation ratios λ $(0.1, 0.3, 0.5, 0.7)$. We find that FedCS is relatively robust to the change of λ. Although FedCS shows a slightly lower learning speed with a small $\lambda = 0.1$, the final test accuracy is the same under different λ, which implies that it is possible to involve fewer participants into FL tasks such that a vast amount of communication costs can be saved. Moreover, the performance fluctuations decrease slightly as λ increases.

D. Influence of Unbalancedness μ in Data Size

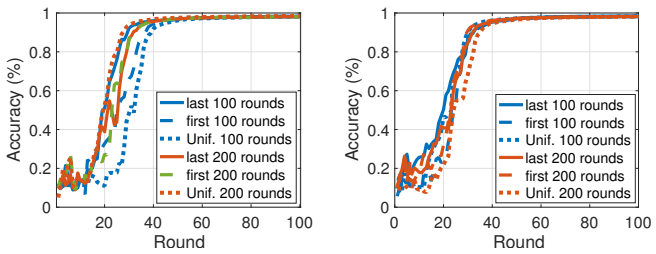
All experiments we performed above are with a balanced split of data sets, where all workers hold the same number of samples. In this experiment, we examine the performance of FedCS on the unbalancedness in the data size [7] . The degree of unbalancedness is defined by

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

where $\mathcal{S}_N = \{|\mathcal{D}_1|, .., |\mathcal{D}_N|\}$. $\mu = 0.1$ implies most of the samples are stored by a small numbers of workers, while $\mu = 1$ means all workers hold the same numbers of samples. We set μ as 0.1, 0.3, 0.5, 0.7 and 1. The convergence trends of FedCS with different μ are illustrated in Fig. 6(b). It can be seen that the unbalancedness does not affect model convergence significantly. We speculate that is because the local models are able to learn properly with IID data, even when the data is unevenly distributed among workers.

E. Influence of Batch Size b

We study how batch size affects the performance of FedCS in this subsection. We perform experiments on the task of Lenet5 on MNIST with IID data, set $\lambda = 0.1$ and $\mu = 1$. The convergence trends of FedCS with different batch sizes (32, 64, 100 and 128) are illustrated in Fig. 6(c). Generally speaking, Properly increasing the batch size will make the model converge faster and smoother. Surprisingly, we observe that increasing the batch size will cause the model convergence to slow down and show large fluctuations. We check the variation of compression ratios during training and figure out that, with



(a) $\mathcal{X}$ with 100 samples (b) $\mathcal{X}$ with 1000 samples

Fig. 7: Convergence curves of FedCS on different input of dictionary learning.

a large batch size, the compression ratio drops more slowly. We speculate that this results in relatively low reconstruction accuracies in the early stage of federated training, and thus a relatively slow and fluctuating model convergence.

F. Influence of Generating Input of Dictionary Learning

In this experiment, we examine how the input of dictionary learning affects FedCS performance. We use MNIST, and generate different sizes of quasi-validation sets (100 and 1000 samples) to train the Lenet5 model for one thousand rounds, and set ζ as 100 and 200. Moreover, we apply three strategies in selecting ζ intermediate models, i.e., selecting first and last ζ models, and uniformly selecting ζ models, from the 1000 models. Fig. 7 illustrates the convergence trends of FedCS under different settings. We find that for a large quasi-validation set (1000 samples), FedCS is more robust such that both ζ and selecting strategies do not affect its convergence significantly. For the quasi-validation set with 100 samples, it can be seen that a large ζ leads to a faster convergence, and selecting last ζ models is the best selection strategy. Importantly, experimental results indicate that even using a small quasi-validation set, FedCS demonstrates satisfactory performances in terms of test accuracy and convergence speed.

VI. Conclusion

In this work, we propose FedCS, a compressive sensing based scheme, which further reduces the communication cost of FL on the basis of Fedavg. We utilize a quasi-validation set to learn a sparse representation dictionary, enabling efficient compression and reconstruction of non-sparse machine learning model parameters. The adaptive compression ratio adjustment technique allows the recovered model to gradually converge and achieve approximate test accuracies to Fedavg with a maximum of 10 times overall compression ratio. In addition, we adopt joint model reconstruction on the server side, and layered compression on the worker side, which guarantees low computational costs of compression. The experimental results suggest that our FedCS outperforms the other three comparison schemes in all three image classification tasks.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China (Grant No. 61972081,62202001), and the Natural Science Foundation OF Shanghai (Grant No. 22ZR1400200), the University Grant Committee of Hongkong (Grant No. 15204820).

References

[1] J. Bernstein, Y. Wang, K. Azizzadenesheli, and A. Anandkumar. signsgd: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, pages 560–569. PMLR, 2018.

[2] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li. Terngrad: Ternary gradients to reduce communication in distributed deep learning. *Advances in neural information processing systems*, 30, 2017.

[3] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. *Advances in neural information processing systems*, 30, 2017.

[4] N. Strom. Scalable distributed dnn training using commodity gpu cloud computing. In *Sixteenth annual conference of the international speech communication association*, 2015.

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

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

[7] F. Sattler, S. Wiedemann, K. Müller, and W. Samek. Robust and communication-efficient federated learning from non-iid data. *IEEE transactions on neural networks and learning systems*, 31(9):3400–3413, 2019.

[8] F. Sattler, S. Wiedemann, K. Müller, and W. Samek. Sparse binary compression: Towards distributed deep learning with minimal communication. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

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

[10] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[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 *International Conference on Artificial Intelligence and Statistics*, pages 2021–2031. PMLR, 2020.

[12] F. Haddadpour, M. M. Kamani, A. Mokhtari, and M. Mahdavi. Federated learning with compression: Unified analysis and sharp guarantees. In *International Conference on Artificial Intelligence and Statistics*, pages 2350–2358. PMLR, 2021.

[13] J. Xu, W. Du, Y. Jin, W. He, and R. Cheng. Ternary compression for communication-efficient federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[14] E. J. Candès, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on information theory*, 52(2):489–509, 2006.

[15] D. L. Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306, 2006.

[16] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020.

[17] E. J. Candes. The restricted isometry property and its implications for compressed sensing. *Comptes rendus mathematique*, 346(9-10):589–592, 2008.

[18] J. A. Tropp and A. C. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on information theory*, 53(12):4655–4666, 2007.

[19] M. Aharon, M. Elad, and A. Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 54(11):4311–4322, 2006.

[20] L. Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.

[21] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[22] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009.