Communication-Efficient Federated Learning Based on Compressed Sensing

Chengxi Li, Gang Li, *Senior Member, IEEE*, and Pramod K. Varshney, *Life Fellow, IEEE*

***Abstract*—In this article, we investigate the problem of federated learning (FL) in a communication-constrained environment of the Internet of Things (IoT), where multiple IoT clients train a global model collectively by communicating model updates with a central server instead of sending raw data sets. To ease the communication burden in IoT systems, several approaches have been proposed for the FL tasks, including sparsification methods and data quantization strategies. To overcome the shortcomings of the existing methods, we propose two new FL algorithms based on compressed sensing (CS) referred to as the CS-FL algorithm and the 1-bit CS-FL algorithm, both of which compress the upstream and downstream data while communicating between the clients and the central server. The proposed algorithms improve upon the existing algorithms by letting the clients send analog and 1-bit data, respectively, to the server after compression with a random measurement matrix. Based on that, in CS-FL and 1-bit CS-FL, the clients update the model locally utilizing the result of sparse reconstruction obtained by iterative hard thresholding (IHT) and binary IHT (BIHT), respectively. Experiments conducted on the MNIST and the Fashion-MNIST data sets reveal the superiority of the proposed algorithm over the baseline algorithms, SignSGD with a majority vote, FL based on sparse ternary compression, and FedAvg.**

***Index Terms*—1-bit quantization, compressed sensing (CS), federated learning (FL), Internet of Things (IoT).**

I. Introduction

NOWADAYS, the Internet of Things (IoT) plays an important role in various smart applications, including smart factory, automatic driving, smart city, and smart ocean [37]–[40], [49]. There have been many recent technological advances that enable the collection of massive amounts of data from different sensors [43], [44] and devices [1], [2] in IoT systems [45], [46]. These data can be utilized to perform tasks such as making inferences regarding a phenomenon of interest, which has facilitated the emergence of data-driven methods such as machine learning (ML) [26]. A large number of ML algorithms operate in a centralized manner, where all the available data sets are collected together at a central server to train a global model such as a neural network. However, the centralized utilization of the data often gives rise to privacy concerns and there is a large communication cost in sending the data sets to the central server, which is not desirable in many practical scenarios [3], [32], [33]. In order to avoid the above drawbacks induced by the sharing of raw data with a central server, while still benefiting from data generated by different IoT clients, the federated learning (FL) paradigm has been proposed, where clients only communicate model updates with the server [4], [27]–[29], [47], [48]. It is worth noting that the FL paradigm is highly pertinent to IoT applications. For example, as shown in [49], in a smart ocean IoT system, it is quite difficult for the underwater networks to set up reliable links to send raw data sets in the presence of considerable fading, which implies that gathering the training data to train models in a centralized manner is impractical and adopting FL is necessary. In [51], in mobile crowdsensing tasks associated with IoT applications, the model owners are stringently regulated by general data protection regulation (GDPR), which prevents them from sharing their data with each other. To collaborate in order to build a better inference model, FL could be adopted. Moreover, based on the fact that the IoT devices such as smartphones are equipped with fast processers, another bonus of FL lies in the full exploitation of the edge processing capabilities of the IoT systems, which alleviates the computation burden at the server. A classical implementation of FL is FedAvg [3]. During each iteration of FedAvg, each participating client calculates the local model update based on its private data set together with the current model, and then transmits the model update to the central server. Upon receiving the local model updates from all the participating clients, the server averages them to obtain the global model update. Then, before the next iteration, the current model is updated at the server based on the global model update and sent back to the participating clients of the next iteration.

As pointed out by McMahan *et al.* [3], the communication cost in the FL framework is fairly high compared with the computation cost. The clients in an FL system typically employ devices that are equipped with fast processors and sufficient computational power so that they can perform computations locally at the client to generate model updates. Although sending model updates instead of raw data sets in FL reduces the amount of transmitted data, the models to be trained such as neural networks still contain a huge number of parameters. Since IoT systems are generally constrained by the communication resources, the communication burden of FL is enormous and the time cost of communication is very high. For example, the convolutional neural network (CNN) adopted

Manuscript received November 17, 2020; revised March 6, 2021; accepted April 9, 2021. Date of publication April 14, 2021; date of current version October 7, 2021. This work was supported in part by National Natural Science Foundation of China under Grant 61790551 and Grant 61925106. *(Corresponding author: Gang Li.)*

Chengxi Li and Gang Li are with the Department of Electronic Engineering, Tsinghua University, Beijing 100084, China (e-mail: gangli@mail.tsinghua.edu.cn).

Pramod K. Varshney is with the Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY 13244 USA.

Digital Object Identifier 10.1109/JIOT.2021.3073112

in [3] has a total number of 1 663 370 parameters. If ten participating clients upload the updates of the model parameters to the server in each iteration with full precision (each parameter uses 32 bits), more than 507 s are needed by each iteration under an upload bit rate of 1 MB/s. From this perspective, high communication cost of FL constrains its efficiency in practical IoT applications. Motivated by the above fact that the most limiting factor of FL in IoT systems is limited communication resources [3], different strategies have been proposed for the FL tasks in order to alleviate the communication overhead without sacrificing much learning performance, which are introduced briefly as follows.

1) *Sparsification Methods:* Strom [5] has proposed to send only those gradients to the central server, whose magnitudes are larger than a threshold, and to store the other gradients in a residual vector to be used in the subsequent iterations. Aji and Heafield [6] presented another approach for gradient sparsification by transmitting a fraction p of gradient entries with the largest magnitudes and storing the unused gradients in the residual vector. However, these sparsification methods only compress the data for upstream communication from the clients to the server and neglect the compression of the data for downstream communication from the server to the clients.
2) *Quantization Strategy:* Different from the sparsification methods, the quantization strategy enables data compression in both directions of communication between the clients and the server [7]–[9]. In [7] and [8], SignSGD with a majority vote is proposed, where the signs of the local updates are sent to the server and fused using a majority vote. In this way, 32 times less communication can be achieved compared with the standard FL algorithm. However, it is also shown that the performance of SignSGD is significantly degraded in the presence of nonindependent and identically distributed (non-IID) training data across different clients [9]. For brevity, SignSGD with a majority vote is referred to as SignSGD hereinafter in this article. In [9], FL based on sparse ternary compression (FL-STC) is proposed, which transmits model updates after sparsification, ternarization, and Golomb encoding before upstream and downstream communications. It is worth noting that although the quantization strategy proposed in [7]–[9] reduces the communication overhead to a large extent, it induces substantial information loss due to very coarse quantization, which results in a large deviation in the actual model updates from the direction of the steepest descent. This degradation of the learning performance is notably enhanced with non-IID training data, which will be shown by the experimental results in Section IV.

Compressed sensing (CS) [10] has been proposed in the literature and is widely used for the reconstruction of high-dimensional sparse signals based on only a small number of compressed measurements, which enables sampling below the Nyquist rate and near-optimal signal reconstruction simultaneously [30], [31], [41], [42]. This is because the information content of sparse signals is much smaller than that indicated by the Nyquist rate. To reconstruct the sparse signals based on compressed measurements, numerous algorithms have been proposed over the years, which could be categorized into linear programming algorithms [10], [15]–[17] and greedy algorithms [13], [18]–[21]. Among these algorithms, the iterative hard thresholding (IHT) algorithm has been extensively studied, which is quite robust to the observation noise and provides guarantees of near-optimal reconstruction. Besides, by using IHT, the required number of iterations only relies on the signal to noise ratio. Due to the above advantages, one of our proposed algorithms in this article utilizes IHT to reconstruct the model updates locally. The detailed algorithm based on IHT will be introduced in Section III.

Furthermore, since the compressed measurements must be quantized in practice, the CS framework has also been extended to include quantization. In [11], it is shown that the signs of the measurements retain ample information to reconstruct the sparse signals. Based on that, CS in an extreme quantization case entitled as 1-bit CS is considered in [11], where the measurements are quantized into 1-bit data by only preserving their signs. Also, a practical algorithm, called binary IHT (BIHT), is proposed in [11], which provides good reconstruction performance of 1-bit CS. 1-bit CS has also been investigated in [34] and [35] and shown to yield excellent performance. Although CS has already become a very useful and popular tool in various domains of signal processing [22]–[25], to the best of our knowledge, it has not yet been applied to FL, which is well worth investigating.

In this article, to overcome the shortcomings of the existing communication-efficient FL algorithms, we exploit the advantages of CS under the FL framework and propose two communication-efficient algorithms based on CS tools for FL in IoT applications. They are referred to as the CS-FL algorithm and the 1-bit CS-FL algorithm, respectively. In the CS-FL algorithm, the local updates are sparsified and compressed into analog compressed measurements, which are then transmitted by the participating clients to the server. Based on them, the server averages the measurements and broadcasts them back to the clients. Based on the received information, the global model update is reconstructed locally by means of the IHT algorithm, with which each client updates the current model synchronously. Different from CS-FL, in the proposed 1-bit CS-FL algorithm, the local updates are sparsified and compressed into 1-bit compressed measurements. All the 1-bit measurements sent from the participating clients are fused at the server by a majority vote and then broadcast back to the clients. BIHT is then employed locally to reconstruct the global model update, with which the current model is updated synchronously at all the clients. It is worth emphasizing that, in both CS-FL and 1-bit CS-FL, the data communicated in both directions between the server and the clients are compressed, which eases the burden for the upstream and the downstream communications. Besides, by using the proposed algorithms, the dominant entries in the model updates can be recovered at the local clients using CS tools for updating the global model so that the actual model updates have strong correlations with the direction of the steepest descent, which guarantees good learning performance of

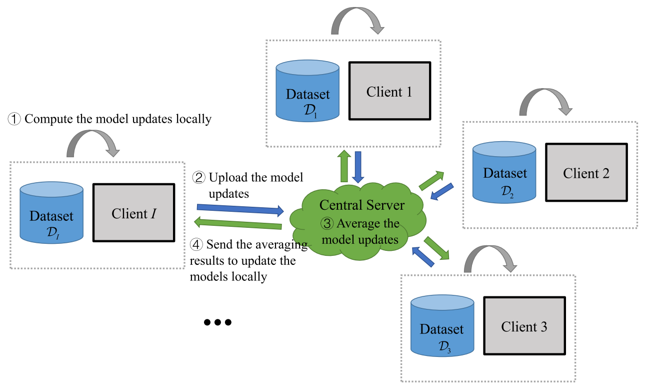

① Compute the model updates locally
② Upload the model updates
③ Average the model updates
④ Send the averaging results to update the models locally
Central Server
Dataset $\mathcal{D}_1$
Client 1
Dataset $\mathcal{D}_2$
Client 2
Dataset $\mathcal{D}_3$
Client 3
Dataset $\mathcal{D}_I$
Client I

Fig. 1. General framework of FL in IoT applications [3].

the proposed algorithms. Experimental results on the MNIST and the Fashion-MNIST data sets show that: 1) both CS-FL and 1-bit CS-FL slightly outperform SignSGD and FL-STC with IID training data; 2) the 1-bit CS-FL algorithm attains superior performance over SignSGD and FL-STC by a large margin under the non-IID case, which demonstrates the robustness of 1-bit CS-FL to non-IID data; and 3) both CS-FL and 1-bit CS-FL achieve better learning performances compared with FedAvg, which verifies the effectiveness of using CS tools to reduce the communication overhead under the FL framework.

The main contributions of this article are given as follows.

1) We incorporate the advantages of the CS tools into the FL framework and propose two new algorithms for the FL tasks, which compress the data for both upstream and downstream communications and, thus, make the system very communication-efficient.
2) We run experiments on two popular data sets to perform image classification tasks using the proposed algorithms. The experimental results demonstrate the superiority of the proposed algorithms. In particular, it is shown that the proposed 1-bit CS-FL algorithm is more robust to non-IID training data across different clients compared with SignSGD and FL-STC.

The remainder of this article is organized as follows. In Section II, the system model and the background are introduced. The proposed algorithms referred to as the CS-FL algorithm and the 1-bit CS-FL algorithm are developed based on the CS tools in Section III. In Section IV, experimental results are provided and discussed to show the superiority of the proposed algorithms. Finally, the article is concluded in Section V.

II. System Model and Background

The FL problem is introduced as follows. A total number of I IoT clients, each of which owns a set $\mathcal{D}_i$ of labeled training data samples, $i = 1, \ldots, I$, want to train a global model for a specific task (e.g., image classification) with the help of a central server without sharing their local data sets [3]. This FL model can be widely applied in IoT systems. For instance, in a smart ocean IoT system [49], the IoT clients are submarines or other devices deployed in the underwater environment, which are able to monitor the undersea activities and to gather the relevant data as training data sets. The central server in the network is a base station that can setup wireless links such as ultrahigh-frequency links with the IoT clients in the underwater environment [49].

Algorithm 1: SignSGD With Majority Vote (Baseline)

```
Input: initial parameters $\mathbf{w}_0$
Output: optimized parameters $\mathbf{w}_{T+1}$
Initialization: All clients are initialized with the same global
model with parameters $\mathbf{w}_0$
For $t = 0, \ldots, T$ do
    A set $\mathcal{I}_t$ of clients are randomly chosen.
    For $i \in \mathcal{I}_t$ in parallel do
        $\mathbf{w}_t^i \leftarrow \mathrm{SGD}_k(\mathbf{w}_t, \mathcal{D}_i)$
        $\mathbf{d}_t^i \leftarrow \mathrm{sign}\left(\mathbf{w}_t^i - \mathbf{w}_t\right)$, push $\mathbf{d}_t^i$ to server
    end
    The server does:
        $\mathbf{d}_t^{glo} \leftarrow \mathrm{sign}\left(\sum_{i\in\mathcal{I}_t} \mathbf{d}_t^i\right)$, push $\mathbf{d}_t^{glo}$ to each client
    For $i \in \{1, 2, \ldots, I\}$ in parallel do
        $\mathbf{w}_{t+1} = \mathbf{w}_t + \beta_t \mathbf{d}_t^{glo}$
    end
end
Return $\mathbf{w}_{T+1}$
```

A general FL framework in IoT applications is shown in Fig. 1, which consists of four steps in each iteration [3]: 1) computing the model updates locally with the private data sets of the clients; 2) uploading the model updates to a central server; 3) averaging the model updates at the server; and 4) sending the averaging results to the clients to update the local models. Under such an FL framework, as introduced in Section I, although FedAvg performs well in terms of the number of iterations needed to attain convergence, the communication cost is significant since the models transmitted between the servers and the clients always contain a huge number of parameters. To deal with this, several FL algorithms based on the quantization strategy have been proposed to improve on FedAvg in order to reduce the communication cost [7]–[9] in both directions between the server and the clients, which include SignSGD [7], [8] and FL-STC [9]. The SignSGD algorithm is presented as Algorithm 1. The specific description of FL-STC is omitted in this article due to space limitations. Interested readers could refer to [9] for more detailed information. Although SignSGD is easy to implement, it is not very robust to non-IID training data across different clients. Besides, SignSGD and FL-STC induce substantial information loss due to very coarse quantization. Next, we will propose two new FL algorithms based on the CS tools and demonstrate their superiority over the existing methods, i.e., FedAvg, SignSGD, and FL-STC.

III. Proposed Algorithms

In this section, the proposed CS-FL algorithm and the 1-bit CS-FL algorithm are presented, whose pseudocodes are provided in Algorithms 2 and 3, respectively. At the beginning of both algorithms, all the clients are initialized with the same model structure and the same model parameters $\mathbf{w}_0$. Servers such as base stations in wireless communications have limited

Algorithm 2: CS-FL Algorithm

```
Input: initial parameters $\mathbf{w}_0$
Output: optimized parameters $\mathbf{w}_{T+1}$
Initialization: All clients are initialized with the same global
model with parameters $\mathbf{w}_0$
For $t = 0, \ldots, T$ do
    A set $\mathcal{I}_t$ of clients are randomly chosen.
    The first phase:
    For $i \in \mathcal{I}_t$ in parallel do
        $\mathbf{w}_t^i \leftarrow \mathrm{SGD}_k(\mathbf{w}_t, \mathcal{D}_i)$
        $\mathbf{h}_t^i \leftarrow \mathbf{w}_t^i - \mathbf{w}_t$
        $\mathbf{s}_t^i \leftarrow \mathrm{spar}\left(\mathbf{h}_t^i\right)$
        $\mathbf{e}_t^i \leftarrow \mathbf{h}_t^i - \mathbf{s}_t^i$
        $\mathbf{y}_t^i \leftarrow \mathbf{A}_t\mathbf{s}_t^i$, push $\mathbf{y}_t^i$ to server
    end
    The server does:
        $\mathbf{y}_t^{glo} \leftarrow \frac{1}{n_t}\sum_{i\in\mathcal{I}_t}\left(\mathbf{y}_t^i\right)$, push $\mathbf{y}_t^{glo}$ to each client
    For $i \in \{1, 2, \ldots, I\}$ in parallel do
        $\tilde{\mathbf{s}}_t \leftarrow \mathrm{IHT}\left(\mathbf{A}_t, \mathbf{y}_t^{glo}\right)$
        $\tilde{\mathbf{w}}_t \leftarrow \mathbf{w}_t + \varpi_t\tilde{\mathbf{s}}_t$
    end
    The second phase:
    For $i \in \mathcal{I}_t$ in parallel do
        $\tilde{\mathbf{w}}_t^i \leftarrow \mathrm{SGD}_k(\tilde{\mathbf{w}}_t, \mathcal{D}_i)$
        $\tilde{\mathbf{h}}_t^i \leftarrow \tilde{\mathbf{w}}_t^i - \tilde{\mathbf{w}}_t$
        $\tilde{\mathbf{r}}_t^i \leftarrow \mathrm{sign}\left(\mathbf{e}_t^i + \tilde{\mathbf{h}}_t^i\right)$, push $\tilde{\mathbf{r}}_t^i$ to server
    end
    The server does:
        $\tilde{\mathbf{r}}_t^{glo} \leftarrow \mathrm{sign}\left(\sum_{i\in\mathcal{I}_t}\tilde{\mathbf{r}}_t^i\right)$, push $\tilde{\mathbf{r}}_t^{glo}$ to each client
    For $i \in \{1, 2, \ldots, I\}$ in parallel do
        $\mathbf{w}_{t+1} = \tilde{\mathbf{w}}_t + \sigma_t\tilde{\mathbf{r}}_t^{glo}$,
    end
end
Return $\mathbf{w}_{T+1}$
```

bandwidth, which makes it possible to let only a few clients upload their local updates simultaneously [12]. Hence, partial client participation is assumed in the proposed algorithm, as done in [3]. Based on that, at the beginning of the tth iteration ($t \geq 0$), a subset $\mathcal{I}_t$ of n_t clients is randomly chosen to be participating clients for that iteration.

In both the CS-FL algorithm and the 1-bit CS-FL algorithm, each iteration is comprised of two phases that are executed sequentially, and all the differences between the two algorithms lie in the first phase. In CS-FL, the uploaded data in the first phase are the analog compressed measurements of the local model updates and the downloaded data in the first phase are the averaged results of the uploaded measurements. In contrast, in 1-bit CS-FL, the uploaded data in the first phase are the 1-bit compressed measurements of the local model updates and the downloaded data are the majority vote results of the uploaded 1-bit data.

The second phases of both algorithms are the same. In both CS-FL and 1-bit CS-FL, the uploaded data in the second phase incorporate 1-bit local model updates and the residual vectors generated by the sparsification in the first phase. Based on that, for both algorithms, the downloaded data in the second phase are the majority vote results of the uploaded 1-bit data. No matter which algorithm is implemented, it can be easily seen that the data communicated in both the upstream and the downstream directions are equally compressed, which implies the communication-efficiency of the proposed algorithms. Next, the implementation of the first phase and the second phase of those two algorithms is discussed in detail.

A. CS-FL Algorithm

1) First Phase: Each participating client conducts gradient descent k times based on its local data set $\mathcal{D}_i$ and the current model $\mathbf{w}_t$ to obtain the updated model $\mathbf{w}_t^i = \mathrm{SGD}_k(\mathbf{w}_t, \mathcal{D}_i)$, $i \in \mathcal{I}_t$, where $\mathcal{I}_t$ is the set containing all the indices of the participating clients in the tth iteration. Based on that, the local model updates can be calculated by $\mathbf{h}_t^i = \mathbf{w}_t^i - \mathbf{w}_t$, $i \in \mathcal{I}_t$.

Next, since it is inefficient to transmit $\mathbf{h}_t^i$ directly to the server, the proposed CS-FL algorithm performs data compression based on CS before transmission. To be more specific, the local updates are sparsified by only preserving the dominant entries of the largest absolute values and dropping the others [6]. The sparsification of $\mathbf{h}_t^i$ as mentioned above can be written as

$$\mathbf{s}_t^i = \mathrm{spar}\left(\mathbf{h}_t^i\right), \ i \in \mathcal{I}_t \tag{1}$$

where $\mathbf{s}_t^i \in \mathbb{R}^N$ denotes the sparsified vector of the local update. To prevent the negative impact on convergence induced by zeroing the small update entries, the dropped values are stored locally in a residual vector [6], which is reused later in the second phase. Based on (1), the residual vector stored at the ith client is given as

$$\mathbf{e}_t^i = \mathbf{h}_t^i - \mathbf{s}_t^i, \ i \in \mathcal{I}_t. \tag{2}$$

Then, $\mathbf{s}_t^i$ is compressed with a random measurement matrix [11] $\mathbf{A}_t \in \mathbb{R}^{M\times N}(M < N)$, which yields the compressed measurements expressed as

$$\mathbf{y}_t^i = \mathbf{A}_t\mathbf{s}_t^i, \ i \in \mathcal{I}_t. \tag{3}$$

Based on (3), the compressed measurements are directly sent to the server and averaged to obtain

$$\mathbf{y}_t^{\mathrm{glo}} = \frac{1}{n_t}\sum_{i\in\mathcal{I}_t}\left(\mathbf{y}_t^i\right) \tag{4}$$

where $n_t = |\mathcal{I}_t|$. After that, $\mathbf{y}_t^{glo}$ is transmitted to all the clients. Then, based on $\mathbf{y}_t^{\mathrm{glo}}$, each client reconstructs the global model update $\tilde{\mathbf{s}}_t$ using the IHT algorithm [13], which can be expressed as

$$\tilde{\mathbf{s}}_t \leftarrow \mathrm{IHT}\left(\mathbf{A}_t, \mathbf{y}_t^{glo}\right). \tag{5}$$

Based on $\tilde{\mathbf{s}}_t$, the global model is updated at each client as

$$\tilde{\mathbf{w}}_t \leftarrow \mathbf{w}_t + \varpi_t\tilde{\mathbf{s}}_t \tag{6}$$

where ϖ_t is the learning rate in the first phase of CS-FL.

2) Second Phase: In the second phase, each participating client first conducts gradient descent k times based on its local data set and the current model $\tilde{\mathbf{w}}_t$. Denote the updated model as $\tilde{\mathbf{w}}_t^i$, $i \in \mathcal{I}_t$. Based on that, the local model updates can be calculated by $\tilde{\mathbf{h}}_t^i = \tilde{\mathbf{w}}_t^i - \tilde{\mathbf{w}}_t$, $i \in \mathcal{I}_t$. Then, the model update $\tilde{\mathbf{h}}_t^i$ is added to the residual vector $\mathbf{e}_t^i$ generated by the sparsification

Algorithm 3: 1-Bit CS-FL Algorithm

```
Input: initial parameters $\mathbf{w}_0$
Output: optimized parameters $\mathbf{w}_{T+1}$
Initialization: All clients are initialized with the same global model with parameters $\mathbf{w}_0$
For $t = 0, \ldots, T$ do
    A set $\mathcal{I}_t$ of clients are randomly chosen.
    The first phase:
    For $i \in \mathcal{I}_t$ in parallel do
        $\mathbf{w}_t^i \leftarrow \mathrm{SGD}_k(\mathbf{w}_t, \mathcal{D}_i)$
        $\mathbf{h}_t^i \leftarrow \mathbf{w}_t^i - \mathbf{w}_t$
        $\mathbf{s}_t^i \leftarrow \mathrm{spar}\left(\mathbf{h}_t^i\right)$
        $\mathbf{e}_t^i \leftarrow \mathbf{h}_t^i - \mathbf{s}_t^i$
        $\mathbf{y}_t^i \leftarrow \mathbf{A}_t\mathbf{s}_t^i$
        $\mathbf{z}_t^i \leftarrow \mathrm{sign}\left(\mathbf{y}_t^i\right)$, push $\mathbf{z}_t^i$ to server
    end
    The server does:
        $\mathbf{z}_t^{glo} \leftarrow \mathrm{sign}\left(\sum_{i\in\mathcal{I}_t} \mathbf{z}_t^i\right)$, push $\mathbf{z}_t^{glo}$ to each client
    For $i \in \{1, 2, \ldots, I\}$ in parallel do
        $\hat{\mathbf{s}}_t \leftarrow \mathrm{BIHT}\left(\mathbf{A}_t, \mathbf{z}_t^{glo}\right)$
        $\bar{\mathbf{w}}_t \leftarrow \mathbf{w}_t + \gamma_t\hat{\mathbf{s}}_t$
    end
    The second phase:
    For $i \in \mathcal{I}_t$ in parallel do
        $\bar{\mathbf{w}}_t^i \leftarrow \mathrm{SGD}_k(\bar{\mathbf{w}}_t, \mathcal{D}_i)$
        $\bar{\mathbf{h}}_t^i \leftarrow \bar{\mathbf{w}}_t^i - \bar{\mathbf{w}}_t$
        $\mathbf{r}_t^i \leftarrow \mathrm{sign}\left(\mathbf{e}_t^i + \bar{\mathbf{h}}_t^i\right)$, push $\mathbf{r}_t^i$ to server
    end
    The server does:
        $\mathbf{r}_t^{glo} \leftarrow \mathrm{sign}\left(\sum_{i\in\mathcal{I}_t} \mathbf{r}_t^i\right)$, push $\mathbf{r}_t^{glo}$ to each client
    For $i \in \{1, 2, \ldots, I\}$ in parallel do
        $\mathbf{w}_{t+1} = \bar{\mathbf{w}}_t + \mu_t\mathbf{r}_t^{glo}$,
    end
end
Return $\mathbf{w}_{T+1}$
```

in the first phase, and then quantized into 1-bit data, which yields

$$\tilde{\mathbf{r}}_t^i = \mathrm{sign}\left(\mathbf{e}_t^i + \tilde{\mathbf{h}}_t^i\right),\ i \in \mathcal{I}_t. \tag{7}$$

Then, the 1-bit data $\{\tilde{\mathbf{r}}_t^i,\ i \in \mathcal{I}_t\}$ are sent by the participating clients to the server, which are aggregated and then fused by majority vote to obtain

$$\tilde{\mathbf{r}}_t^{glo} = \mathrm{sign}\left(\sum_{i\in\mathcal{I}_t} \tilde{\mathbf{r}}_t^i\right). \tag{8}$$

Next, the global update vector $\tilde{\mathbf{r}}_t^{glo}$ is transmitted back to all the clients and the model is updated locally by

$$\mathbf{w}_{t+1} = \tilde{\mathbf{w}}_t + \sigma_t\tilde{\mathbf{r}}_t^{glo} \tag{9}$$

where σ_t is the learning rate of the second phase in CS-FL.

B. 1-Bit CS-FL Algorithm

1) First Phase: Similar to the first phase of the CS-FL algorithm, in 1-bit CS-FL, each participating client first performs gradient descent k times utilizing its local data set together with the current model $\mathbf{w}_t$, which yields the updated model $\mathbf{w}_t^i = \mathrm{SGD}_k(\mathbf{w}_t, \mathcal{D}_i)$, $i \in \mathcal{I}_t$. After that, the local model updates are derived locally as $\mathbf{h}_t^i = \mathbf{w}_t^i - \mathbf{w}_t$, $i \in \mathcal{I}_t$.

Next, different from CS-FL, the proposed 1-bit CS-FL algorithm lets the clients transmit 1-bit compressed measurements based on 1-bit CS. To be more specific, the local updates $\mathbf{h}_t^i$ are sparsified as introduced in (1) to obtain the sparsified vector $\mathbf{s}_t^i \in \mathbb{R}^N$ of the local update. Similar to (2), the residual vector $\mathbf{e}_t^i$ is derived and stored at the ith client accordingly. Then, $\mathbf{s}_t^i$ is compressed with a random measurement matrix [11] $\mathbf{A}_t \in \mathbb{R}^{M\times N}(M < N)$, which yields the compressed measurements $\mathbf{y}_t^i$ by means of (3). After that, the compressed measurements are quantized into 1-bit data by only preserving their signs, and the 1-bit compressed measurements are given as

$$\mathbf{z}_t^i = \mathrm{sign}\left(\mathbf{y}_t^i\right),\ i \in \mathcal{I}_t \tag{10}$$

where $\mathrm{sign}(\cdot)$ denotes the elementwise sign function. Next, the 1-bit compressed measurements obtained by (10) are sent to the server by the participating clients. Upon receiving these 1-bit measurements, the server conducts a majority vote for the fusion, which can be written as

$$\mathbf{z}_t^{glo} = \mathrm{sign}\left(\sum_{i\in\mathcal{I}_t} \mathbf{z}_t^i\right) \tag{11}$$

and the fusion result $\mathbf{z}_t^{glo} \in \{-1, 1\}^M$ is transmitted back to all the clients. Then, based on $\mathbf{z}_t^{glo}$, each client reconstructs the global model update $\hat{\mathbf{s}}_t$ by means of the BIHT algorithm [11], which can be expressed as

$$\hat{\mathbf{s}}_t = \mathrm{BIHT}\left(\mathbf{A}_t, \mathbf{z}_t^{glo}\right). \tag{12}$$

Based on $\hat{\mathbf{s}}_t$, the global model at each client is updated as

$$\bar{\mathbf{w}}_t = \mathbf{w}_t + \gamma_t\hat{\mathbf{s}}_t \tag{13}$$

where γ_t is the learning rate in the first phase of 1-bit CS-FL.

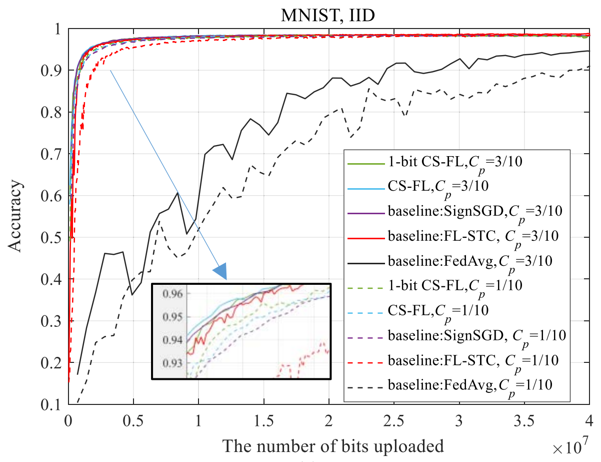
2) Second Phase: The second phases of 1-bit CS-FL and CS-FL are exactly the same. In the second phase of 1-bit CS-FL, each participating client first obtains the local model update after conducting gradient descent k times based on its local data set and the current model $\bar{\mathbf{w}}_t$. Then, the sum of the model update and the residual vector is quantized into 1-bit data and sent by the participating clients to the server. After those 1-bit data are aggregated and fused by a majority vote at the server, the global update vector is sent back to all the clients so that the model can be updated locally at each client.

Remark 1: During each iteration of both 1-bit CS-FL and CS-FL, both the participating clients and the nonparticipating clients have to download global model updates from the server synchronously and utilize them to update their current model. It is worth noting that as an alternative the server can update the model and then transmit the updated model instead of the model updates to the clients [3]. In that case, only the participating clients need to download the updated model, but the server has to send a complete set of model parameters, which is less communication-efficient than the proposed algorithms that only require the server to transmit compressed data to the clients.

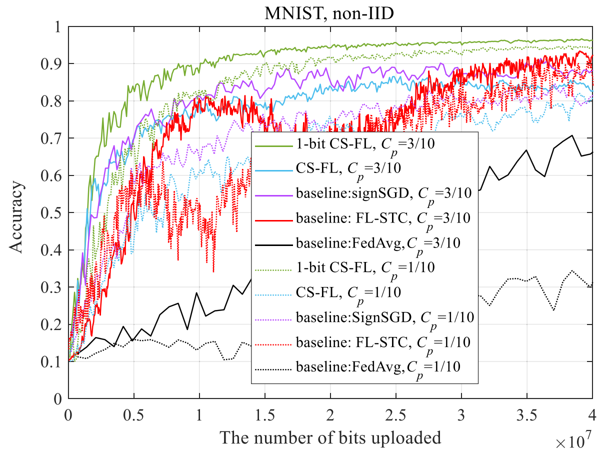
Remark 2 (1-bit CS-FL Versus CS-FL): Contrary to CS, which only emphasizes the reduction of the sampling rate, 1-bit CS takes both the sampling rate and the bit depth of the compressed measurements into account [11]. Since CS and 1-bit CS represent two different ways of data compression and reconstruction, it is worthwhile to explore which one between CS-FL and 1-bit CS-FL attains better learning performance under the same communication overhead. Note that the same communication cost of CS-FL and 1-bit CS-FL can be attained by appropriately choosing two different compression ratios M/N for the two algorithms, as done in the simulation part. In Section IV, the experimental results will reveal that CS-FL and 1-bit CS-FL perform similarly with IID training data while 1-bit CS-FL significantly outperforms CS-FL under the non-IID case, which indicates that 1-bit CS-FL is more robust to non-IID training data and is thus more practical. To justify it, some insights behind this result will be provided in Section IV. Besides, it is also worth pointing out that, in practice, the 1-bit quantizer in 1-bit CS-FL can be easily reduced to a comparator testing for values that are positive or negative and, therefore, is very simple and efficient to realize, which means 1-bit CS-FL is also easily implementable.

Remark 3 (Rationale Behind the Majority Vote in the Compressed Domain in the First Phase of 1-Bit CS-FL): Since only a few clients participate in each iteration and s_t^i can be highly sparsified, $\mathbf{s}_t = \sum_{i\in\mathcal{I}_t} \mathbf{s}_t^i/n_t$ is sparse even though the sparsity patterns of the individually sparsified vectors $\{\mathbf{s}_t^i, i \in \mathcal{I}_t\}$ differ. Therefore, based on the CS theory, $\mathbf{s}_t$ can be reconstructed from $\text{sign}(\mathbf{A}_t\mathbf{s}_t)$ based on BIHT, where $\mathbf{A}_t \in \mathbb{R}^{M\times N}$ is the random measurement matrix. This is equivalent to reconstructing $\mathbf{s}_t$ from $\text{sign}(\sum_{i\in\mathcal{I}_t} \mathbf{A}_t\mathbf{s}_t^i/n_t)$. Note that the entries in $\mathbf{A}_t$ are independently drawn from the standard normal distribution, which means that each entry in $\mathbf{A}_t\mathbf{s}_t^i$ follows the Gaussian distribution $\mathcal{N}(0, \|\mathbf{s}_t^i\|_2^2)$. Also note that the values of $\|\mathbf{s}_t^i\|_2^2$ at different clients whose local optimizers share the same learning rate are very close to each other. Then, it can be deduced that all of the entries in $\mathbf{A}_t\mathbf{s}_t^i$ follow similar distributions across different clients and the absolute values of the entries in $\mathbf{A}_t\mathbf{s}_t^i$ do not deviate much from each other. Based on that, approximating $\text{sign}(\sum_{i\in\mathcal{I}_t} \mathbf{A}_t\mathbf{s}_t^i/n_t)$ by $\text{sign}(\sum_{i\in\mathcal{I}_t} \text{sign}(\mathbf{A}_t\mathbf{s}_t^i)/n_t)$, $\mathbf{s}_t$ can be reconstructed from $\text{sign}(\sum_{i\in\mathcal{I}_t} \text{sign}(\mathbf{A}_t\mathbf{s}_t^i)/n_t)$ approximately. Noting that $\text{sign}(\sum_{i\in\mathcal{I}_t} \text{sign}(\mathbf{A}_t\mathbf{s}_t^i)/n_t)$ is the result of majority vote of the 1-bit compressed measurements at the server in the first phase of 1-bit CS-FL, it is intuitive that the global model update can be reconstructed from the result of the majority vote of the 1-bit compressed measurements received by the server.

Remark 4 (Sparsification Methods): In the proposed algorithms, in the first phase, the local updates are sparsified by dropping the entries that have small absolute values. For the sake of convenience, this method of sparsification can be referred to as absolute sparsification. An alternative for sparsification is to use a random mask (RaM) to sparsify the local update vectors at the participating clients so that all of them share a common sparsity pattern. It is obvious that the sum of the sparsified local updates, i.e., $\mathbf{s}_t = (1/n_t)\sum_{i\in\mathcal{I}_t} \mathbf{s}_t^i$, obtained by RaM is more sparse than that obtained by absolute sparsification, since absolute sparsification does not guarantee $\{\mathbf{s}_t^i, i \in \mathcal{I}_t\}$ to share the same sparsity pattern. However, RaM may not preserve the dominant entries with the largest values in the local update vectors. Experiments will be conducted in Section IV to demonstrate the superiority of absolute sparsification adopted by the proposed algorithms.



(a)



(b)

Fig. 2. Test accuracy versus the number of bits uploaded for CS-FL, 1-bit CS-FL, SignSGD, FL-STC, and FedAvg with varying participation rates C_p, where a CNN is trained on the MNIST data set. (a) IID case. (b) Non-IID case.

IV. Experiments

In this section, the performances of 1-bit CS-FL and CS-FL are evaluated for image classification tasks and compared with the existing FL algorithms, including SignSGD, FedAvg, and FL-STC to show their superiority.

Data Sets and Models: CNNs are trained for the image classification tasks on two popular data sets, i.e., the MNIST data set [14] and the Fashion-MNIST data set [36]. MNIST data set contains 60000 training images and 10 000 test images of handwritten digits from 0 to 9. Fashion-MNIST data set contains 60 000 training images and 10 000 test images of ten different classes of fashion items. The CNNs trained on MNIST and Fashion-MNIST have a total number of 21 840 parameters and 29 034 parameters, respectively.

Implementation Details: The FL system is composed of a central server and I clients. In the following experiments, unless specified, I is fixed at 10 to simulate the scenarios of the horizontally FL to business (H2B) [50], where there are a handful of clients in the network. All the training samples are allocated to the clients, ensuring that each client has the same number of samples. The central server has no training data and is capable of communicating with the local clients. The test data set containing 10 000 test samples is used to test the classification accuracy of the trained model. The SGD optimizer with momentum 0.5 is adopted, and each participating client implements a single-step ($k = 1$) SGD of local batch size 200 to obtain the local model update in both phases of an iteration. In these experiments, both IID and non-IID training data are considered. For the IID case, all the training samples are shuffled and partitioned randomly so that each client receives equal number of samples with the same distribution. For the non-IID case, the training samples are first sorted by their labels and then divided into $Q \times I$ segments of equal size, where each segment mainly contains samples of a single label. Each client obtains Q segments randomly, which indicates that most of the clients have access to samples of only Q classes. This operation enables us to verify the robustness of the proposed algorithm to highly non-IID data. When running experiments on MNIST and Fashion-MNIST under the non-IID case, we set $Q = 2$ and $Q = 8$, respectively. When the proposed algorithms are implemented, the sparsity level is set as $p = 0.005$ and the compression ratio in 1-bit CS-FL is fixed at $r_{\text{com}} = M/N = 0.1$. Under each experimental setting, we test the learning performance of the proposed algorithms on different learning rates, and the optimal learning rate is obtained which is then employed for further experiments. In the first phase of each iteration, the elements in the random measurement matrix are generated independently from the standard normal distribution. The test accuracy, which is the ratio of the number of correctly classified samples in the test data set to the total size of the test data set, quantifies the performance of different algorithms. Under each setting, the presented results are obtained over three runs to reduce the impact of the randomness of training with CNNs.

A. 1-Bit CS-FL Versus CS-FL

To investigate the differences between 1-bit CS-FL and CS-FL, we perform experiments using both algorithms on the MNIST and the Fashion-MNIST data sets. In Fig. 2, for 1-bit CS-FL and CS-FL, we plot the test accuracy versus the number of bits uploaded by each participating client, which is attained by the CNN trained on MNIST under different participation rates C_p, i.e., the ratio of the number of participating clients in each iteration to the total number of the clients. Accordingly, the test accuracy values attained after a certain amount of data is uploaded by each participating client are presented in Fig. 3. For a fair comparison, the same number of bits are uploaded per iteration for both algorithms. To this end, considering each analog measurement sent by the clients in CS-FL requires 32 bits, while each measurement in 1-bit CS-FL only requires 1 bit, the compression ratio r_{com} of CS-FL is reduced by a factor of $\times 32$ compared with that of 1-bit CS-FL. With non-IID training data, the learning rates of CS-FL are $\{\varpi_t = 0.1, \sigma_t = 0.0003\}$ and $\{\varpi_t = 0.1, \sigma_t = 0.0005\}$ for $C_p = 0.1$ and $C_p = 0.3$, respectively. The learning rates of 1-bit CS-FL are $\{\gamma_t = 0.1, \mu_t = 0.0003\}$ and $\{\gamma_t = 0.1, \mu_t = 0.0005\}$ for $C_p = 0.1$ and $C_p = 0.3$, respectively. With IID training data, the learning rates adopted by CS-FL are $\{\varpi_t = 0.2, \sigma_t = 0.002\}$ for $C_p = 0.1$ and $C_p = 0.3$. The learning rates of 1-bit CS-FL are $\{\gamma_t = 0.2, \mu_t = 0.002\}$ for $C_p = 0.1$ and $C_p = 0.3$. From Fig. 3, it can be seen that, with IID training data, CS-FL slightly outperforms 1-bit CS-FL (within 0.16%). In contrast, with non-IID data, 1-bit CS-FL attains superior performance than CS-FL (more than 10%), which indicates that the former suffers less from the non-IID data. Similarly, from Fig. 2(a), it can be observed that the curves of 1-bit CS-FL almost overlap those of CS-FL under the same setting with IID training data. However, from Fig. 2(b), we can easily see that 1-bit CS-FL converges faster to higher values of test accuracy than CS-FL. From another perspective, to attain the same test accuracy, 1-bit CS-FL requires the local clients to upload less data to the central

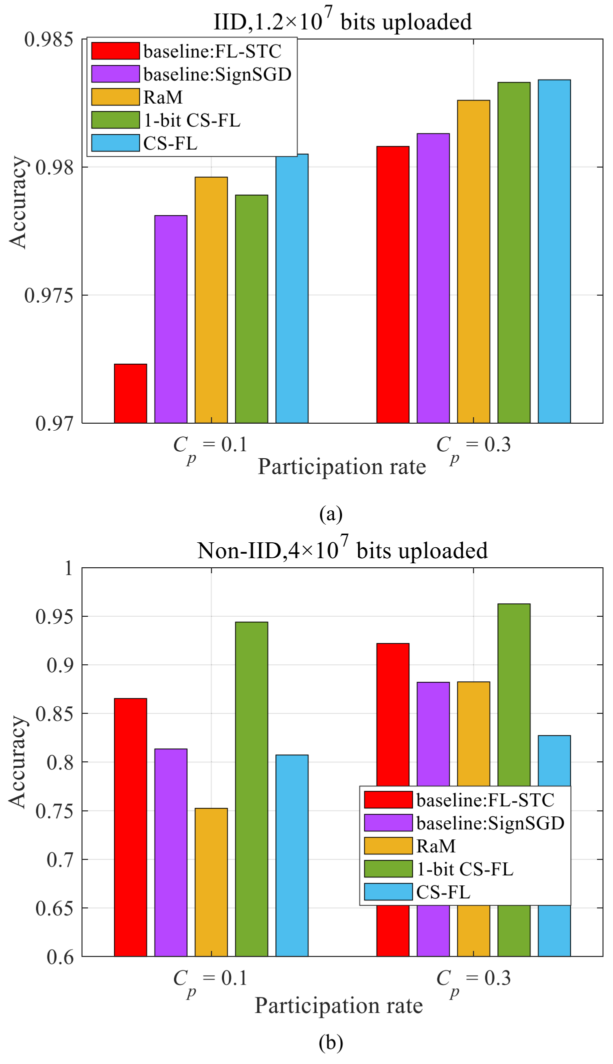


Fig. 3. Test accuracy of different algorithms with a CNN trained on the MNIST data set. (a) IID case. (b) Non-IID case.

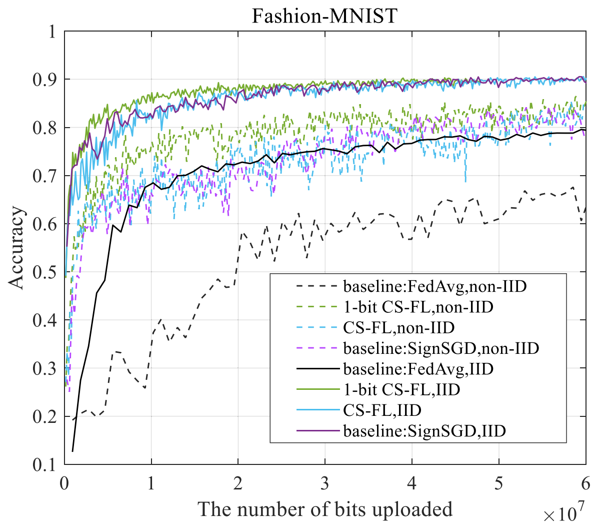


Fig. 4. Test accuracy versus the number of bits uploaded for CS-FL, 1-bit CS-FL, FedAvg, and SignSGD, where a CNN is trained on the fashion-MNIST data set.

server compared with CS-FL, which means the time cost of communication reduces for 1-bit CS-FL with fixed communication resources. As stated in Section I, the communication cost is the main cost of FL compared with the computation cost [3], which implies that 1-bit CS-FL is more efficient than CS-FL. The subsequent experimental results can be analyzed from this perspective similarly.

Likewise, in Fig. 4, we plot the test accuracy versus the number of bits uploaded by each participating client when the CNN is trained on the more challenging Fashion-MNIST data set, where CS-FL and 1-bit CS-FL are, respectively, implemented with the participation rate fixed at $C_p = 0.1$. The learning rates of CS-FL are $\{\varpi_t = 0.1, \sigma_t = 0.0005\}$ and $\{\varpi_t = 0.2, \sigma_t = 0.001\}$ under the non-IID and IID cases, respectively. The learning rates of 1-bit CS-FL are $\{\gamma_t = 0.1, \mu_t = 0.0005\}$ and $\{\gamma_t = 0.2, \mu_t = 0.001\}$ under the non-IID and IID cases, respectively. The test accuracy values achieved when a certain amount of data is uploaded by each participating client are shown in Fig. 5. From Figs. 4 and 5, it can be easily seen that, for both the IID case and the non-IID case, 1-bit CS-FL converges faster than CS-FL and the former attains higher test accuracy with a certain amount of data uploaded to the central server.

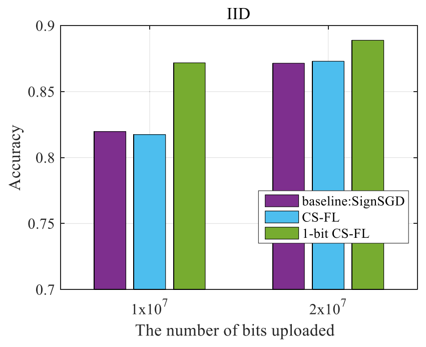
The above results reveal that transmitting low-precision compressed measurements (1-bit data in 1-bit CS-FL) is more effective than transmitting high-precision compressed data (the analog measurements in CS-FL), under the same volume of data transmission. This is because, for equal communication overhead, a small number of compressed measurements with unnecessarily high precision are utilized in CS-FL while a large number of low-precision measurements are available in 1-bit CS-FL. The reduction of the sampling rate has more negative impact on the training performance than the reduction of measurement precision. In other words, more severe information loss is induced by compression and sampling in CS-FL compared with that in 1-bit CS-FL. Hence, the sparse reconstruction in CS-FL is less accurate, and the model updates reconstructed in 1-bit CS-FL are closer to the direction of the steepest descent. Based on that, for the considered problem in this article, it is not a bad idea to sacrifice some of the precision of the compressed measurements in exchange for a higher sampling rate in order to enhance the overall training performance, as done in 1-bit CS-FL.

To sum up, the results in Section IV-A validate the following.

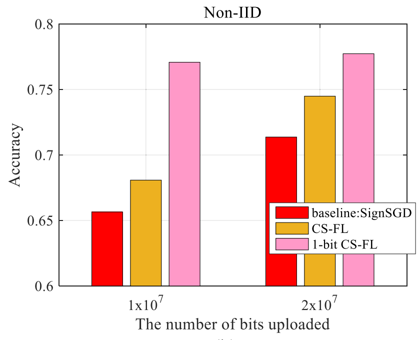
1) Both CS-FL and 1-bit CS-FL perform well under the IID case.
2) Compared with CS-FL, 1-bit CS-FL is more robust to non-IID training data.

B. Comparison Between the Proposed Algorithms and the Baseline Methods

To compare the performances of the proposed algorithms and the baseline methods, including SignSGD [7], [8], FedAvg [3], and FL-STC [9], we apply the baseline methods on the MNIST and the Fashion-MNIST data sets. For the implementation of FL-STC, the parameter p of STC is fixed at 0.5. Different learning rates are tested for the implementation of each method and the best one is found and applied under each setting. When the CNN is trained on MNIST, for the baseline methods, the test accuracy versus the number of bits uploaded by each participating client under different participation rates C_p is shown in Fig. 2 and the test accuracy values are given in Fig. 3 that are attained after a certain amount of data are uploaded to the central server by each participating client. It can be easily observed from Figs. 2 and 3 that under the IID case, both CS-FL and 1-bit CS-FL slightly outperform SignSGD and FL-STC and significantly outperform FedAvg. Under the non-IID case, 1-bit CS-FL outperforms the three baseline methods by a large margin. CS-FL behaves worse than SignSGD and FL-STC while it maintains superior performance over FedAvg. When the CNN is trained on Fashion-MNIST, we depict the test accuracy versus the number of bits uploaded by each participating client in Fig. 4, where the participation rate is fixed at $C_p = 0.1$. In this case, we have found that FL-STC suffers from severe oscillations and could hardly converge so we omit the experimental results with FL-STC and only compare the learning performances of the proposed algorithms with two baseline methods, i.e., FedAvg and SignSGD. The test accuracy values achieved when a certain amount of data are uploaded by each participating client are presented in Fig. 5. From Figs. 4 and 5, it can be clearly seen that 1-bit CS-FL attains superior performance than SignSGD under both the IID case and the non-IID case by a significant margin, while there is only a negligible performance gap between CS-FL and SignSGD. It is also obvious that both the proposed algorithms and SignSGD outperform FedAvg by a large margin, which benefits from the strategies adopted by CS-FL, 1-bit CS-FL, and SignSGD to alleviate the communication burden. From Figs. 2 and 4, we can also observe that the test accuracy curves oscillate more severely under the non-IID case than that under the IID case, no matter which one of the proposed algorithms and the baseline algorithms is employed. This complies with our intuition



(a)



(b)

Fig. 5. Test accuracy attained by CS-FL, 1-bit CS-FL and SignSGD after uploading a certain amount of data to the central server, where a CNN is trained on the fashion-MNIST data set. (a) IID case. (b) Non-IID case.

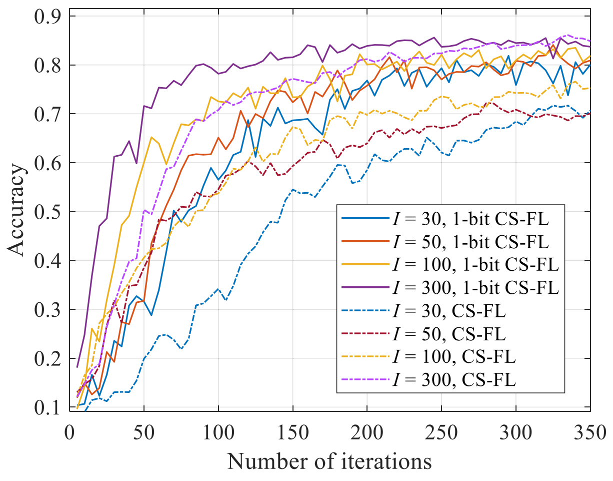


Fig. 6. Test accuracy versus the number of iterations for the proposed algorithms, where a CNN is trained on the MNIST data set.

that non-IID data results in instability of training and learning under the FL framework.

From the above experimental results, it is obvious that 1-bit CS-FL is the most robust to the non-IID training data among the tested methods, whose learning performance suffers from the least degradation under that case. This is due to the fact that dominant entries in the model updates can be recovered accurately when 1-bit CS-FL is performed and the actual model updates have strong correlations with the direction of the steepest descent, which is not true for FL-STC and SignSGD that quantize the entries in the update vectors of different magnitudes to the same level. In practice, the distributions of training data may vary widely across different clients, which implies the high practical value of 1-bit CS-FL, especially under complex scenarios. As expected, the methods adopting different strategies to reduce the communication overhead of FL, i.e., 1-bit CS-FL, CS-FL, FL-STC, and SignSGD all outperform FedAvg that transmits model updates without any further processing, which demonstrates the effectiveness of data compression before communication under the FL framework.

C. Sparsification Methods

It has been demonstrated in Section IV-A and IV-B that 1-bit CS-FL is more robust to non-IID training data. Let us gain more insights into this algorithm by conducting some additional experiments. Next, we verify the superiority of the absolute sparsification method that we adopt in the 1-bit CS-FL algorithm. To this end, we substitute the absolute sparsification of 1-bit CS-FL by RaM and provide its test accuracies attained by training the CNN on MNIST under the same settings in Fig. 3. It can be seen that under the IID case, both methods achieve similar results. Besides, as expected, the performance of both methods suffers from the non-IID training data. However, under the non-IID case, the performance with RaM degrades more severely than that with absolute sparsification adopted by the proposed 1-bit CS-FL. This is due to the fact that under the non-IID case, random sparsification discards the dominant entries in the local update vector with a higher probability, which results in an inevitably large deviation in the actual model updates from the direction of the steepest descent.

D. Influence of the Number of the Clients on the Learning Performances of the Proposed Algorithms

The above experiments were conducted with a handful of clients, which is the case encountered by H2B [50]. Next, to investigate the potential of the proposed algorithms for horizontally FL to consumers (H2C), where more clients are incorporated into the learning system, we run additional experiments on the MNIST data set using CS-FL and 1-bit CS-FL with non-IID training data. The learning rates found in Section IV-A are adopted for both algorithms. In Fig. 6, we depict the test accuracy versus the number of iterations in the network, where the number of clients varies and the participation rate is fixed at $C_p = 0.1$. It can be easily seen that, with more clients, the trained models can achieve higher test accuracy after the same number of iterations. Although increasing the number of clients implies that each client obtains less training samples in our experiments, more clients in the network contribute to more effective utilization of the decentralized computational resources and thus result in better learning performance. It is worth noting that the performance gain achieved by increasing the number of clients is at the cost of heavier communication burden. From this perspective, in practical FL applications, the effectiveness of the proposed FL algorithms depends on the available communication resources and the number of clients in the network. Besides, Fig. 6 also shows that 1-bit CS-FL is more robust to non-IID training data than CS-FL with varying numbers of clients, which complies with our observations in Figs. 2–5.

V. Conclusion

In this article, the problem of FL with limited communication resources was studied. We proposed two new algorithms, named CS-FL and 1-bit CS-FL, where the clients compress and reconstruct the sparsified local updates by CS and 1-bit CS, respectively. In the proposed algorithms, the analog compressed measurements or 1-bit compressed measurements are fused at the central server. Based on that, global updates can be reconstructed at the local clients so as to update the current model. The proposed algorithms have three advantages over existing methods: 1) the data communicated in both upstream and downstream directions are compressed, which improves its communication efficiency; 2) through experiments of image classification tasks on MNIST and Fashion-MNIST, CS-FL, and 1-bit CS-FL slightly outperform SignSGD and FL-STC and significantly outperform FedAvg with IID training data; and 3) it was also demonstrated that 1-bit CS-FL outperforms the existing methods, including FedAvg, FL-STC, and SignSGD in terms of the robustness to non-IID data. Some future research avenues include providing more theoretical justifications about the performance of the proposed algorithms and investigating the application of the CS tools to more complex FL scenarios such as decentralized FL without using a central server.

References

[1] J. Mills, J. Hu, and G. Min, "Communication-efficient federated learning for wireless edge intelligence in IoT," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5986–5994, Jul. 2020.

[2] W. Y. B. Lim *et al.*, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 2031–2063, 3rd Quart., 2020.

[3] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Stat.*, 2017, pp. 1273–1282.

[4] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.

[5] N. Strom, "Scalable distributed DNN training using commodity GPU cloud computing," in *Proc. INTERSPEECH*, 2015, pp. 1488–1492.

[6] A. F. Aji and K. Heafield, "Sparse communication for distributed gradient descent," in *Proc. Conf. Empirical Methods Nat. Lang. Process.*, Copenhagen, Denmark, Sep. 2017, pp. 440–445.

[7] J. Z. J. Bernstein, K. Azizzadenesheli, and A. Anandkumar, "signSGD with majority vote is communication efficient and fault tolerant," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019, pp. 1–6.

[8] Y. W. J. Bernstein, K. Azizzadenesheli, and A. Anandkumar, "SignSGD: Compressed optimisation for non-convex problems," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 559–568.

[9] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-i.i.d. data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 9, pp. 3400–3413, Sep. 2020.

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

[11] L. Jacques, J. N. Laska, P. T. Boufounos, and R. G. Baraniuk, "Robust 1-bit compressive sensing via binary stable embeddings of sparse vectors," *IEEE Trans. Inf. Theory*, vol. 59, no. 4, pp. 2082–2102, Apr. 2013.

[12] A. M. A. Reisizadeh, H. Hassani, A. Jadbabaie, and R. Pedarsani, "FedPAQ: A communication-efficient federated learning method with periodic averaging and quantization," in *Proc. 23rd Int. Conf. Artif. Intell. Stat.*, 2020, pp. 2021–2031.

[13] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressed sensing," *Appl. Comput. Harmon. Anal.*, vol. 27, no. 3, pp. 265–274, Nov. 2009.

[14] Y. LeCun. (1998). *The MNIST Database of Handwritten Digits*. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[15] E. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Inf. Theory*, vol. 52, pp. 489–509, Feb. 2006.

[16] E. Candès, J. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Commun. Pure Appl. Math.*, vol. 59, no. 8, pp. 1207–1223, 2006.

[17] E. Candès and J. Romberg, "Quantitative robust uncertainty principles and optimally sparse decompositions," *Found. Comput. Math*, vol. 6, no. 2, pp. 227–254, 2006.

[18] D. Needell and R. Vershynin, "Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit," *Found. Comput. Math.*, vol. 9, pp. 317–334, Jun. 2009.

[19] D. Needell and R. Vershynin, "Signal recovery from incomplete and inaccurate measurements via regularized orthogonal matching pursuit," *IEEE J. Sel. Topics Signal Process.*, vol. 4, no. 2, pp. 310–316, Apr. 2010.

[20] W. Dai and O. Milenkovic, "Subspace pursuit for compressive sensing signal reconstruction," *IEEE Trans. Inf. Theory*, vol. 55, no. 5, pp. 2230–2249, May 2009.

[21] D. Needell and J. A. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples," *Appl. Comput. Harmon. Anal.*, vol. 26, no. 3, pp. 301–321, 2009.

[22] M. Ke, Z. Gao, Y. Wu, X. Gao, and R. Schober, "Compressive sensing-based adaptive active user detection and channel estimation: Massive access meets massive MIMO," *IEEE Trans. Signal Process.*, vol. 68, pp. 764–779, Jan. 2020.

[23] I. Taghavi, M. F. Sabahi, and F. Parvaresh, "High resolution compressed sensing radar using difference set codes," *IEEE Trans. Signal Process.*, vol. 67, no. 1, pp. 136–148, Jan. 2019.

[24] X. Wang, G. Li, Y. Liu, and M. G. Amin, "Two-level block matching pursuit for polarimetric through-wall radar imaging," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 3, pp. 1533–1545, Mar. 2018.

[25] J. Han, G. Li, and X.-P. Zhang, "Refocusing of moving targets based on low-bit quantized SAR data via parametric quantized iterative hard thresholding," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 56, no. 3, pp. 2198–2211, Jun. 2020.

[26] X. Wu, J. Zhang, and F. Wang, "Stability-based generalization analysis of distributed learning algorithms for big data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 3, pp. 801–812, Mar. 2020.

[27] F. Sattler, K. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Aug. 24, 2020, doi: 10.1109/TNNLS.2020.3015958.

[28] H. Zhu and Y. Jin, "Multi-objective evolutionary federated learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 4, pp. 1310–1322, Apr. 2020.

[29] Y. Chen, X. Sun, and Y. Jin, "Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 10, pp. 4229–4238, Oct. 2020.

[30] Z. Dong and W. Zhu, "Homotopy methods based on l_0-norm for compressed sensing," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 4, pp. 1132–1146, Apr. 2018.

[31] C. Guo and Q. Yang, "A neurodynamic optimization method for recovery of compressive sensed signals with globally converged solution approximating to l_0 minimization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 7, pp. 1363–1374, Jul. 2015.

[32] R. Fierimonte, S. Scardapane, A. Uncini, and M. Panella, "Fully decentralized semi-supervised learning via privacy-preserving matrix completion," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 11, pp. 2699–2711, Nov. 2017.

[33] B. Liu, Z. Ding, and C. Lv, "Distributed training for multi-layer neural networks by consensus," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 5, pp. 1771–1778, May 2020.

[34] G. Joseph, S. Kafle, and P. K. Varshney, "One-bit compressed sensing using generative models," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Barcelona, Spain, 2020, pp. 3437–3441.

[35] S. Kafle, V. Gupta, B. Kailkhura, T. Wimalajeewa, and P. K. Varshney, "Joint sparsity pattern recovery with 1-b compressive sensing in distributed sensor networks," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 5, no. 1, pp. 15–30, Mar. 2019.

[36] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017. [Online]. Available: arXiv:1708.07747.

[37] L. Lou, Q. Li, Z. Zhang, R. Yang, and W. He, "An IoT-driven vehicle detection method based on multisource data fusion technology for smart parking management system," *IEEE Internet Things J.*, vol. 7, no. 11, pp. 11020–11029, Nov. 2020.

[38] Ş. Kolozali *et al.*, "Observing the pulse of a city: A smart city framework for real-time discovery, federation, and aggregation of data streams," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2651–2668, Apr. 2019.

[39] M. Mohammadi, A. Al-Fuqaha, M. Guizani, and J. Oh, "Semisupervised deep reinforcement learning in support of IoT and smart city services," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 624–635, Apr. 2018.

[40] W. Xu *et al.*, "The design, implementation, and deployment of a smart lighting system for smart buildings," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 7266–7281, Aug. 2019.

[41] P. Wang, J. Fang, H. Duan, and H. Li, "Compressed channel estimation for intelligent reflecting surface-assisted millimeter wave systems," *IEEE Signal Process. Lett.*, vol. 27, pp. 905–909, May 2020.

[42] Y. Du *et al.*, "Block-sparsity-based multiuser detection for uplink grant-free NOMA," *IEEE Trans. Wireless Commun.*, vol. 17, no. 12, pp. 7894–7909, Dec. 2018.

[43] Y. Liao, X. Shen, and Y. Zhu, "Distributed detection fusion with nonideal channels under Monte Carlo framework," in *Proc. 20th Int. Conf. Inf. Fusion*, 2017, pp. 1–8.

[44] X. Shen, S. Liu, and P. K. Varshney, "Sensor selection for nonlinear systems in large sensor networks," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 50, no. 4, pp. 2664–2678, Oct. 2014.

[45] A. L. Diedrichs, F. Bromberg, D. Dujovne, K. Brun-Laguna, and T. Watteyne, "Prediction of frost events using machine learning and IoT sensing devices," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4589–4597, Dec. 2018.

[46] O. B. Sezer, E. Dogdu, and A. M. Ozbayoglu, "Context aware computing, learning, and big data in Internet of Things: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 1–27, Feb. 2018.

[47] Y. Zhan, P. Li, Z. Qu, D. Zeng, and S. Guo, "A learning-based incentive mechanism for federated learning," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6360–6368, Jul. 2020.

[48] C. Zhou, A. Fu, S. Yu, W. Yang, H. Wang, and Y. Zhang, "Privacy-preserving federated learning in fog computing," *IEEE Internet Things J.*, vol. 7, no. 11, pp. 10782–10793, Nov. 2020.

[49] D. Kwon, J. Jeon, S. Park, J. Kim, and S. Cho, "Multiagent DDPG-based deep learning for smart ocean federated learning IoT networks," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9895–9903, Oct. 2020.

[50] L. Lyu, H. Yu, and Q. Yang. (2020). *Threats to Federated Learning: A Survey*. [Online]. Available: http://arxiv. org/abs/2003.02133

[51] W. Y. B. Lim *et al.*, "Hierarchical incentive mechanism design for federated machine learning in mobile networks," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9575–9588, Oct. 2020.

Chengxi Li received the B.S. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2018. She is currently pursuing the Ph.D. degree with Tsinghua University, Beijing, China.

Her main research interests are sparse signal processing, distributed signal processing, and distributed machine learning.

Gang Li (Senior Member, IEEE) received the B.S. and Ph.D. degrees in electronic engineering from Tsinghua University, Beijing, China, in 2002 and 2007, respectively.

Since July 2007, he has been with the Faculty of Tsinghua University, where he is currently a Professor with the Department of Electronic Engineering. From 2012 to 2014, he visited Ohio State University, Columbus, OH, USA, and Syracuse University, Syracuse, NY, USA. He has authored or coauthored more than 160 journal and conference papers. He has authored *Advanced Sparsity-Driven Models and Methods for Radar Applications* (London, U.K.: IET–SciTech Publishing, 2020). His research interests include radar signal processing, distributed signal processing, remote sensing, and information fusion.

Prof. Li is an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING. He was the Guest Editor of the *IET Radar, Sonar & Navigation* for the Special Issue on "Innovative Radar Detection, Tracking and Classification for Small UAVs as an Emerging Class of Targets."

Pramod K. Varshney (Life Fellow, IEEE) was born in Allahabad, India, in 1952. He received the B.S. degree (Hons.) in electrical engineering and computer science and the M.S. and Ph.D. degrees in electrical engineering from the University of Illinois at Urbana–Champaign, Champaign, IL, USA, in 1972, 1974, and 1976, respectively, and the Doctor of Engineering degree (Hons.) from Drexel University, Philadelphia, PA, USA, in 2014.

From 1972 to 1976, he held teaching and research assistantships with the University of Illinois. Since 1976, he has been with Syracuse University, Syracuse, NY, USA, where he is currently a Distinguished Professor of Electrical Engineering and Computer Science and the Director of the Center for Advanced Systems and Engineering. As the Director of CASE, he is responsible for technology transition of university expertise to make economic impact in the high-tech economy of New York. He served as an Associate Chair of the Department from 1993 to 1996. He is also an Adjunct Professor of Radiology with Upstate Medical University, Syracuse. He has published extensively. He has authored *Distributed Detection and Data Fusion* (New York, NY, USA: Springer-Verlag, 1997). His current research interests include distributed sensor networks and data fusion, detection and estimation theory, wireless communications, image processing, radar signal processing, physical-layer security, and machine learning.

Dr. Varshney received the 1981 ASEE Dow Outstanding Young Faculty Award. In 2000, he received the Third Millennium Medal from the IEEE and the Chancellor's Citation for exceptional academic achievement at Syracuse University. He also received the IEEE 2012 Judith A. Resnik Award, the ECE Distinguished Alumni Award from the University of Illinois in 2015, and the ISIF's Yaakov Bar-Shalom Award for a Lifetime of Excellence in Information Fusion in 2018. He was the Guest Editor of the Special Issue on Data Fusion of the PROCEEDINGS OF THE IEEE in 1997. He is on the Editorial Board of the *Journal of Advances in Information Fusion* and has served on the Editorial Boards of the IEEE TRANSACTIONS ON SIGNAL PROCESSING and the *IEEE Signal Processing Magazine*. He was the President of International Society of Information Fusion in 2001. He was a James Scholar, a Bronze Tablet Senior, and a fellow while at the University of Illinois. He is a member of Tau Beta Pi.