Ternary Compression for Communication-Efficient Federated Learning

Jinjin Xu, Wenli Du, Yaochu Jin, *Fellow, IEEE*, Wangli He, *Senior Member, IEEE*, and Ran Cheng, *Member, IEEE*

***Abstract*— Learning over massive data stored in different locations is essential in many real-world applications. However, sharing data is full of challenges due to the increasing demands of privacy and security with the growing use of smart mobile devices and Internet of thing (IoT) devices. Federated learning provides a potential solution to privacy-preserving and secure machine learning, by means of jointly training a global model without uploading data distributed on multiple devices to a central server. However, most existing work on federated learning adopts machine learning models with full-precision weights, and almost all these models contain a large number of redundant parameters that do not need to be transmitted to the server, consuming an excessive amount of communication costs. To address this issue, we propose a federated trained ternary quantization (FTTQ) algorithm, which optimizes the quantized networks on the clients through a self-learning quantization factor. Theoretical proofs of the convergence of quantization factors, unbiasedness of FTTQ, as well as a reduced weight divergence are given. On the basis of FTTQ, we propose a ternary federated averaging protocol (T-FedAvg) to reduce the upstream and downstream communication of federated learning systems. Empirical experiments are conducted to train widely used deep learning models on publicly available data sets, and our results demonstrate that the proposed T-FedAvg is effective in reducing communication costs and can even achieve slightly better performance on non-IID data in contrast to the canonical federated learning algorithms.**

***Index Terms*— Communication efficiency, deep learning, federated learning, ternary coding.**

Manuscript received January 28, 2020; revised July 25, 2020 and October 6, 2020; accepted November 23, 2020. This work was supported in part by the National Natural Science Foundation of China under Basic Science Center Program 61988101, in part by the National Natural Science Fund for Distinguished Young Scholars under Grant 61725301, in part by the International (Regional) Cooperation and Exchange under Project 1720106008, in part by the National Natural Science Foundation of China under Major Program 61590923, and in part by the China Scholarship Council under Grant 201906745025. *(Corresponding authors: Wenli Du; Yaochu Jin.)*

Jinjin Xu, Wenli Du, and Wangli He are with the Key Laboratory of Advanced Control and Optimization for Chemical Processes, Ministry of Education, East China University of Science and Technology, Shanghai 200237, China, and also with the Shanghai Institute of Intelligent Science and Technology, Tongji University, Shanghai 200092, China (e-mail: jin.xu@mail.ecust.edu.cn; wldu@ecust.edu.cn; wanglihe@ecust.edu.cn).

Yaochu Jin is with the Department of Computer Science, University of Surrey, Guildford GU2 7XH, U.K., and also with the Shanghai Institute of Intelligent Science and Technology, Tongji University, Shanghai 200092, China (e-mail: yaochu.jin@surrey.ac.uk).

Ran Cheng is with the Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China (e-mail: ranchengcn@gmail.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2020.3041185.

Digital Object Identifier 10.1109/TNNLS.2020.3041185

I. Introduction

THE number of Internet of things (IoTs) and deployed smart mobile devices in factories has dramatically grown over the past decades, generating massive amounts of data stored distributively every moment. Meanwhile, recent achievements in deep learning [1], [2], such as AlphaGo [3], rely heavily on the knowledge stored in big data. Naturally, the utilization of the vast amounts of data stored in decentralized client devices will significantly benefit human life and industrial production. However, training deep learning models using distributed data is challenging, and uploading private data to the cloud is controversial, due to limited network bandwidths, limited computational resources, and regulations on security and privacy preservation, e.g., the European General Data Protection Regulation (GDPR) [4].

Many research efforts have been devoted to distributed machine learning [5]–[8], which focuses on model training on multiple machines to alleviate the computational burden caused by large data volumes. These methods have achieved promising performance by splitting big data into smaller sets, partitioning models or sharing updates of the model to accelerate the training process with the help of data parallelism [5], model parallelism [5], [9], and parameter server [10]–[13], among others. Correspondingly, weights optimization strategies for multiple machines have also been proposed. For example, Zhang *et al.* [14] introduce asynchronous mini-batch stochastic gradient descent (ASGD) algorithm on multi-GPU devices for deep neural networks (DNNs) training and achieved more than three times speedup on four GPUs than on a single one without loss of precision. Distributed machine learning algorithms for multiple data centers located in different regions have been studied in [15]. However, little attention has been paid to the data security and the impact of data distribution on the performance.

To protect data security and privacy while training models on distributed devices, an interesting framework for training a global model while storing the private data locally, known as federated learning [16]–[18], has been proposed. This approach makes it possible to extract knowledge from data distributed on local devices without uploading private data to a server. Fig. 1 illustrates a simplified workflow and an application scenario in the process industry. Several extensions have been introduced to the standard federated learning system. Zhao *et al.* [19] have observed weight divergence caused by extreme data distributions and proposed a method of sharing a

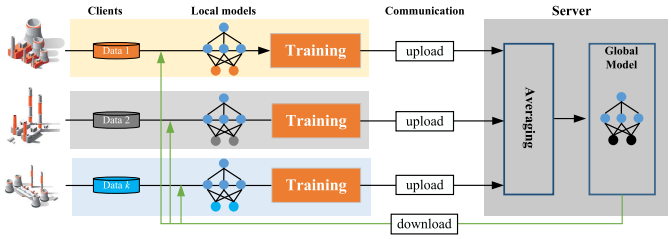

Fig. 1. Illustrative example of federated learning in process industry. The branch factories train local models on local clients and send the trained local models to the server for aggregation to obtain the global model. Then, all factories download the aggregated global model and continue to update the model using their local data.

small amount of data with other clients to enhance the performance of federated learning algorithms. Wang *et al.* [20] have proposed an adaptive federated learning system under a given computational budget based on a control algorithm that balances the client update and global aggregation, and analyzed the convergence bound of the distributed gradient descent. Recent comprehensive overviews of federated learning can be found in [21] and [22], and design ideas, challenges, and future research directions of federated learning on massively mobile devices are presented in [20], [23], [24].

Since the importance of privacy protection and data security is gradually increasing, federated learning has received increasing attention in deep learning, although many challenges remain with respect to data distribution and communication costs.

1) *Data distribution*: Data generated by different clients, e.g., factories, may be unbalanced and not subject to the independent and identical distribution hypothesis, which is known as unbalanced and non-IID data sets, respectively.
2) *Communication costs*: Federated learning is influenced by the rapidly growing depth of model and the amount of multiply-accumulate operations (MACs) [25]. This is due to the needed massive communication costs for uploading and downloading, and the average upload and download speeds are asymmetric, e.g., the mean mobile download speed is 26.36 Mbps while the upload speed is 11.05 Mbps in Q3-Q4 2017 in the U.K. [26].

Clearly, high communication costs are one of the main reasons hindering distributed and federated training. Although the research on model compression was originally not meant to reduce the communication costs, it has become a source of inspiration for communication-efficient distributed learning. Neural network pruning is an early method for model compression proposed in [27]. Parameter pruning and sharing in [27] and [28], low-rank factorization [29], transferred/compact convolutional filters [30], and knowledge distillation in [31]–[33] are some main ideas reported in the literature. A multiobjective evolutionary federated learning framework is presented in [34], which aims to simultaneously maximize the learning performance and minimize the model complexity to reduce communication costs. Wang *et al.* [35] significantly enhance the communication efficiency by uploading significant updates of the local models only. Recently, an layer-wise asynchronous model update approach has been proposed in [36] to reduce the number of the parameters to be transmitted.

Gradient quantization has been proposed to accelerate data parallelism distributed learning [37]; gradient sparsification [38], and gradient quantization [39]–[42] have been developed to reduce the model size; an efficient federated learning algorithm using sparse ternary compression (STC) has been proposed [43], which is robust to non-IID data and communication-efficient on both upstream and downstream communications. However, since STC is a model compression method after local training is completed, the quantization process is not optimized during the training.

To the best of our knowledge, most existing federated learning methods emphasize on the application of full-precision models or streamline the models after the training procedure on the clients is completed, rather than simplifying the model during the training. Therefore, deploying federated learning in real-world systems such as the widely used IoT devices in the process industry is somehow difficult. To address this issue, we focus on model compression on clients during the training to reduce the required computational resources at the inference stage and the communication costs of federated learning. The main contributions of this article are summarized as follows:

1) A ternary quantization approach is introduced into the training and inference of the local models in the clients. The quantized local models are well suited for making inferences in network edge devices.
2) A ternary federated learning protocol is presented to reduce the communication costs between the clients and the server, which reduces both upstream and downstream communication costs. The quantization of the weights of the local models can further enhance privacy preservation since it makes the reverse engineering of the model parameters more difficult.
3) Theoretical analyses of the proposed algorithm regarding the convergence and unbiasedness of the quantization, as well as a reduction in weight divergence are provided. The performance of the algorithm is empirically verified using a deep feedforward network (MLP) and a deep residual network (ResNet) on two widely used data sets, namely MNIST [44] and CIFAR10 [45].

The remainder of this article is organized as follows. In Section II, we briefly review the standard federated learning protocol and several widely used network quantization approaches. Section III proposes a method to quantize the local models on the clients in federated learning systems, called federated trained ternary quantization (FTTQ). On the basis of FTTQ, a ternary federated learning protocol that reduces both upstream and downstream communication costs is presented. In Section IV, theoretical analyses of the proposed algorithm are given. Experimental settings and results are presented in Section V to compare the proposed algorithms with centralized learning, the vanilla federated learning algorithm, and a communication-efficient variant of federated learning. Finally, conclusions are drawn and future directions are suggested in Section VI.

II. Background and Methods

In this section, we first introduce some preliminaries of the standard federated learning workflow and its basic formulations. Subsequently, the definitions and main features of popular ternary quantization methods are presented.

A. Federated Learning

It is usually assumed that the data used by a distributed learning algorithm belongs to the same feature space, which may not be always true in federated learning. As illustrated in Fig. 1, the data sets used for training local models are collected by independent clients [23].

In this work, we assume supervised learning is used for training the local models. Before training, the global model parameters θ are downloaded from the server and deployed on client k, which will be trained based on local data set D_k consisting of training pairs $(x_i,\ y_i)$, $(i = 1, 2, 3, \ldots, |D_k|)$, resulting in the kth local model. The loss function for training the local model on client k is J_k

$$J_k(\theta) = \frac{1}{|D_k|}\sum_{i}^{|D_k|} l(x_i, y_i; \theta). \tag{1}$$

We assume there are N clients whose data are stored independently, and the aim of federated learning is to minimize the global loss J. Therefore, the global loss function of the federated learning system can be defined as

$$J(\theta) = \sum_{k=1}^{\lambda N} \frac{|D_k|}{\sum_{k=1}^{\lambda N}|D_k|} J_k(\theta) \tag{2}$$

where λ is the proportion of the clients participating in the current round of training.

Theoretically, the participation ratio λ in (2) is calculated by the number of participating clients and the total number of clients. Additionally, local epochs E, i.e., the number of iterations for training the local models per round, and local batch size B (i.e., the mini-batch size used in training the local models) are also important hyperparameters [16] in federated learning. Pilot studies have been performed to determine these hyperparameters. However, the results are not reported here due to space limit.

It is noted that communication costs are heavily dependent on the amount of information to be transferred between the server and the clients, and the dominating factor in this procedure is the size of the parameters in the local and global models. One important requirement for communication-efficient federated learning is that both upstream and downstream communications need to be compressed [43]. Note also that the performance of federated learning may dramatically drop due to poor data distribution.

B. Quantization

Quantization improves energy and space efficiency of deep networks by reducing the number of bits per weight [46]. This is done by mapping the parameters in a continuous space to a quantization discrete space, which can greatly reduce model redundancy and save memory overhead. For example, the weights of a full-precision model are compressed into 0 and 1 in [39]. However, the quantization of network weights may cause performance degeneration. To alleviate this problem, a ternary weight network (TWN) is proposed in [40], which quantizes the full-precision weights θ into ternary θ^t (consisting of -1, 0, $+1$) and calculates a scaling factor α^* as follows:

$$\alpha^*, \theta^{t*} = \arg\min_{\alpha,\theta^t} \|\theta - \alpha\theta^t\|_2^2 \tag{3}$$

where α^* and θ^{t*} are the optimal solutions, and $\theta \approx \alpha^*\theta^{t*}$. A ternary quantization with layer-wise thresholds that can approximate the above optimal solution can be obtained by

$$\theta_l^t = \begin{cases} +1, & \theta_l > \Delta_l \\ 0, & |\theta_l| \le \Delta_l \\ -1, & \theta_l < -\Delta_l \end{cases} \tag{4}$$

where the θ_l and θ_l^t are the full-precision and quantized weights of the lth layer, and $\theta = \{\theta_1, \theta_2, \ldots, \theta_l, \ldots\}$, $\theta^t = \{\theta_1^t, \theta_2^t, \ldots, \theta_l^t, \ldots\}$, respectively, and Δ_l is the layer-wise threshold. Li *et al.* [40] provide a rule of thumb to calculate the optimal threshold $\Delta_l^* = (0.7/d^2)\sum_i^{d^2}(|\theta_l^i|)$, where d is the dimension of matrix θ_l, and the optimal $\alpha_l^* = (1/|I_{\Delta_l}|)\sum_i^{|I_{\Delta_l}|}|\theta_i|$, where $I_{\Delta_l} = \{i \mid \theta_i > \Delta_l\}$.

The scaling factor α is empirically shown to be helpful to reduce the Euclidean distance between θ and θ^{t*} [40]. However, the scaling factor α^* of TWN is calculated by rule of thumb, which may go wrong in some cases. To overcome this problem and improve the performance of quantized deep networks, Zhu *et al.* [47] propose a trained ternary quantization algorithm (TTQ). In TTQ, two quantization factors (a positive factor $w_{p,l}$ and a negative factor $w_{n,l}$) are adopted to scale the ternary weights in each layer.

The workflow of TTQ is illustrated in Fig. 2, where the normalized full-precision weights are quantized by Δ_l, $w_{p,l}$ and $w_{n,l}$ with full-precision gradients

$$\theta_l^t = \begin{cases} +1 \times w_{p,l}, & \theta_l > \Delta_l \\ 0, & |\theta_l| \le \Delta_l \\ -1 \times w_{n,l}, & \theta_l < -\Delta_l \end{cases} \tag{5}$$

and TTQ adopts the following heuristic method to calculate Δ_l:

$$\Delta_l = \text{threshold} \times \max(|\theta_l|) \tag{6}$$

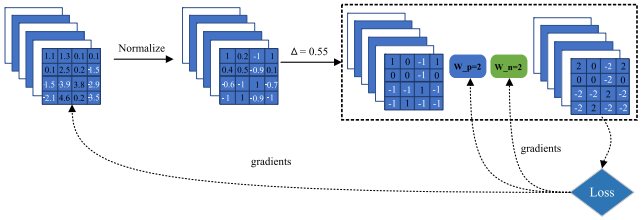

Fig. 2. Example of how the TTQ algorithm works. First, normalized full-precision weights and biases are quantized to $\{-1, 0, +1\}$ by the given layer-wise threshold. Second, positive and negative quantization factors are used to scale the ternary weights. Finally, the calculated gradients are back-propagated to each layer. The right part in the dotted rectangle represents the inference stage.

where threshold is a constant factor determined by experience and $\Delta = \{\Delta_1, \Delta_2, \ldots, \Delta_l, \ldots\}$.

III. Proposed Algorithm

In this section, we first propose a FTTQ to reduce the energy consumption (computational resources) required for each client during inference and the upstream and downstream communication costs. Subsequently, a ternary federated averaging protocol (T-FedAvg for short) is suggested.

A. Federated Trained Ternary Quantization

One challenge remains when threshold-based quantization is to be implemented in a federated learning system. Since the sparsity of the quantized local models may greatly differ from client to client, the global model will be biased toward the local model having the maximum weight range, if the same threshold is used for all clients. To address this issue, we start by normalizing the weights to $[-1, 1]$

$$\theta^s = g(\theta) \tag{7}$$

where θ^s is the normalized weight matrix, g is a normalization function, which normalizes a vector into the range of $[-1, 1]$. However, magnitude imbalance [48] may be introduced when normalizing θ of an entire network at once instead of layer by layer, resulting in a significant precision loss, since many small weights may be pushed to zero. Therefore, we normalize the weights layer by layer.

For simplicity, we drop the subscript l of the quantization factors, thresholds, and weights hereafter. Thus, we calculate the quantization threshold Δ according to the normalized weights as follows:

$$\Delta = T_k \times \max(|\theta^s|) \tag{8}$$

$$T_k = \begin{cases} 0.05 + 0.01 \times \text{rand}(0, 1), & \text{if rand}(0, 1) > 0.5 \\ 0.05 + 0.01 \times \dfrac{k}{N}, & \text{if rand}(0, 1) \leq 0.5 \end{cases} \tag{9}$$

where T_k is a parameter for the kth client, and rand$(0, 1)$ is an operation that generates uniformly distributed random numbers within [0, 1). Note that we introduce randomness into T_k to make it difficult to reverse the weights. We limit the randomness to a controlled range to avoid a significant loss in performance.

However, according to (7) and (8), we can find that the thresholds in all layers are mostly the same since the maximum absolute value of the normalized θ^s is 1 in most layers, which is not good for privacy preservation. Inspired from [40], we adopt the following approach to determine the threshold according to the sparsity of the weights:

$$\Delta = \frac{T_k}{d^2} \sum_{i}^{d^2} (|\theta_i^s|) \tag{10}$$

where d is the dimension of matrix θ_i^s and Δ is calculated layer-wise.

Clearly, the threshold obtained by (10) is influenced by the layer sparsity, making it more difficult to reverse the local weights. Nevertheless, it can still be seen as an extension of (8) since the following relationship holds:

$$\begin{aligned} \Delta &= T_k \times \frac{1}{d^2} \sum_{i}^{d^2} (|\theta_i^s|) \leq T_k \times \frac{1}{d^2} (d^2 \times \max |\theta^s|) \\ &\leq T_k \times \frac{1}{d^2} (d^2 \times 1) \leq T_k \end{aligned} \tag{11}$$

which means that Δ is adaptive, and Δ equals the optimal value suggested in [40] if we set the value of T_k to 0.7.

Subsequently, several operations are taken to achieve layer-wise weight quantization:

$$\text{mask}(\theta^s) = \varepsilon(|\theta^s| - \Delta) \tag{12}$$

$$I^t = \text{sign}(\text{mask} \odot \theta^s) \tag{13}$$

$$\theta^t = w^q \times I^t \tag{14}$$

where ε is the step function, $\odot$ is the Hadamard product, mask(θ^s) is a function whose elements will be stepped to 1 if the absolute value exceeds Δ, w^q is a layer-specific quantization factor to be trained together with weights of the local model layer by layer, and I^t is the quantized ternary weights. Consequently, mask(θ^s) can be rewritten as a union of a positive index matrix I_p and a negative index matrix I_n:

$$I_p = \{i \mid \theta_i^s > \Delta\} \tag{15}$$

$$I_n = \{i \mid \theta_i^s < -\Delta\}. \tag{16}$$

From (14), we can see that different from the standard TTQ, we adopt one quantization factor instead of two quantization factors in each layer. The main reason is that as we theoretically prove in Section IV-A, the two quantization factors w_p, w_n in TTQ will converge to the same absolute value. Consequently, use of one quantization factor can further reduce the communication and computation costs. Additionally, a single quantization factor may also be able to alleviate weight divergence frequently observed in federated learning [19].

After quantizing the whole network, the loss function can be calculated and the errors can be back-propagated. However, the quantization will cause the derivative $(\partial\theta_t/\partial\theta)$ to be infinity or zero, resulting in difficulties training the whole network, since

$$\frac{\partial J}{\partial \theta} = \frac{\partial J}{\partial \theta^t} \cdot \frac{\partial \theta^t}{\partial \theta}. \tag{17}$$

To avoid this problem, the gradients of w^q and the latent full-precision model are calculated in the following way, as suggested in [47]:

$$\frac{\partial J}{\partial w^q} = \sum_{i \in I_p} \frac{\partial J}{\partial \theta_i^t} \tag{18}$$

$$\frac{\partial J}{\partial \theta} = \begin{cases} 1 \times \dfrac{\partial J}{\partial \theta^t}, & |\theta| \leq \Delta \\ w^q \times \dfrac{\partial J}{\partial \theta^t}, & \text{otherwise.} \end{cases} \tag{19}$$

The new update rule is summarized in Algorithm 3. Consequently, FTTQ significantly reduces the size of the updates transmitted to the server, thus reducing the costs of upstream communications. However, the costs of the downstream communications will not be reduced if no additional

Algorithm 1: Federated Trained Ternary Quantization (FTTQ)

Input: Full-precision parameters θ and quantization vector w^q, loss function l, dataset D with sample pairs $(x_i, y_i), i = \{1, 2, ..., |D|\}$, learning rate η

Output: Quantized model θ^t

for $(x_i, y_i) \in D$ **do**

$\theta^s \leftarrow g(\theta)$

$mask(\theta^s) \leftarrow \varepsilon(|\theta^s| - \Delta)$

$I^t = \text{sign}(mask(\theta^s) \odot \theta^s)$

$\theta^t \leftarrow w^q \times I^t$

$J \leftarrow l_i(x_i, y_i; \theta^t)$

$w^q \leftarrow w^q + \eta \frac{\partial J}{\partial w^q}$

$\theta \leftarrow \theta + \eta \frac{\partial J}{\partial \theta}$

end

Return θ^t (including w^q, I^t)

Fig. 3. Federated trained ternary quantization (FTTQ).

measures are taken, since the weights of the global model become real-values again after aggregation. To address this issue, a ternary federated learning protocol is presented in Section III-B.

B. Ternary Federated Averaging

The proposed ternary federated averaging protocol is illustrated in Fig. 4. The local models are normalized and quantized during the training. The quantized model parameters and the quantization factors are then uploaded to the server. Then, the server aggregates all local models to obtain the global model. Finally, the server quantizes the global model again and sends the quantized global model back to all clients. The details are described below.

1) Upstream: Let $\mathbb{K} = \{1, 2, \ldots, |\lambda N|\}$ be the set of indices of the participating clients, where λ is the participation ratio and N is the total number of the clients in the federated learning system. The local normalized full-precision and quantized weights of client $k \in \mathbb{K}$ are denoted by θ_k^s and θ_k^t, respectively. We upload the trained θ_k^t (w^q and I^t) to the server. Note that at the inference stage, only the quantized model is employed for prediction.

2) Downstream: At each communication round, the server will convert each uploaded quantized local model into a continuous model and aggregate them into a global model. Then, we have two strategies for send the global model back to all clients. In Strategy I, the server will also quantize the global model with a threshold $\Delta^S = 0.05 \times \max(|\theta_r|)$ and two server quantization factors $w_p^S = (1/|I_p^S|) \sum_i^{|I_p^S|} (|\theta_r^i|)$, $w_n^S = (1/|I_n^S|) \sum_i^{|I_n^S|} (|\theta_r^i|)$, where r denotes the rth round, I_p^S and I_n^S are calculated according to (15) and (16), respectively. Then, the server broadcasts the quantized global model to the clients. In Strategy II, by contrast, the server will send the full-precision model θ_{r+1} to all clients if the quantized global model crashes. The quantized global model is considered to be collapsed if the performance drop is greater than 3%.

By quantizing the local and global models, our method is able to reduce communications in both upload and download phases, which brings a major advantage when deploying DNNs for resource-constrained devices. The difference between Strategy I and Strategy II lies in the downstream communication costs, and Strategy I is more communication-efficient. According to our experiments, we observed that the server has implemented Strategy I all the time when MLP is adopted as the learning model. Specifically, the clients quantize the local 32-bits full-precision networks to 2-bit quantized models and push the 2-bit network weights and the quantization factor to the server, and then download the quantized global model from the server at the end. For example, if we configure a federated learning environment involving 20 clients and a global model that requires 25 MB of storage space, the total communication costs of the standard federated learning is about 1 GB per round (upload and download). By contrast, our method reduces the costs to 65 MB per round (upload and download), which is about 1/16 of the standard method. The overall workflow of the proposed ternary federated protocol is summarized in Algorithm 5.

IV. Theoretical Analysis

In this section, we first theoretically demonstrate the convergence property of the two quantization factors w_p, w_n in TTQ, followed by a proof of unbiasedness of FTTQ and T-FedAvg. Finally, we provide a theoretical proof of a reduced weight divergence on non-IID data in T-FedAvg.

A. Convergence of Quantization Factors in TTQ

In this subsection, we first prove that the two quantization factors used in TTQ will converge to the same absolute value, which motivates us to use a single quantization factor. To theoretically prove the convergence, we at first introduce the following assumption.

Assumption 1: The elements in the normalized full-precision θ are uniformly distributed between -1 and 1,

$$\forall \theta_i \in \theta, \theta_i \backsim U(-1, 1). \tag{20}$$

Then, we have the following proposition.

Proposition 1: Given a one-layer online gradient system, each element of its parameters is initialized with a symmetric probability distribution centered at 0, e.g., $\theta_i \backsim U(-1, 1)$, which is quantized by TTQ with two iteratively adapted factors w_p, w_n and a fixed threshold Δ, then we have

$$\lim_{e \to +\infty} w_p = \lim_{e \to +\infty} w_n \tag{21}$$

where e is the training epoch and $w_p, w_n, \Delta > 0$.

Proof 1: The converged w_p^* and w_n^* can be regarded as the optimal solution of the quantization factors, which can reduce the Euclidean distance between the full-precision weights θ and the quantized weights θ^t, which is equal to $w_p I_p - w_n I_n$. Then, we have

$$w_p^*, w_n^* = \arg\min_{w_p, w_n} \|\theta - w_p I_p + w_n I_n\|_2^2 \tag{22}$$

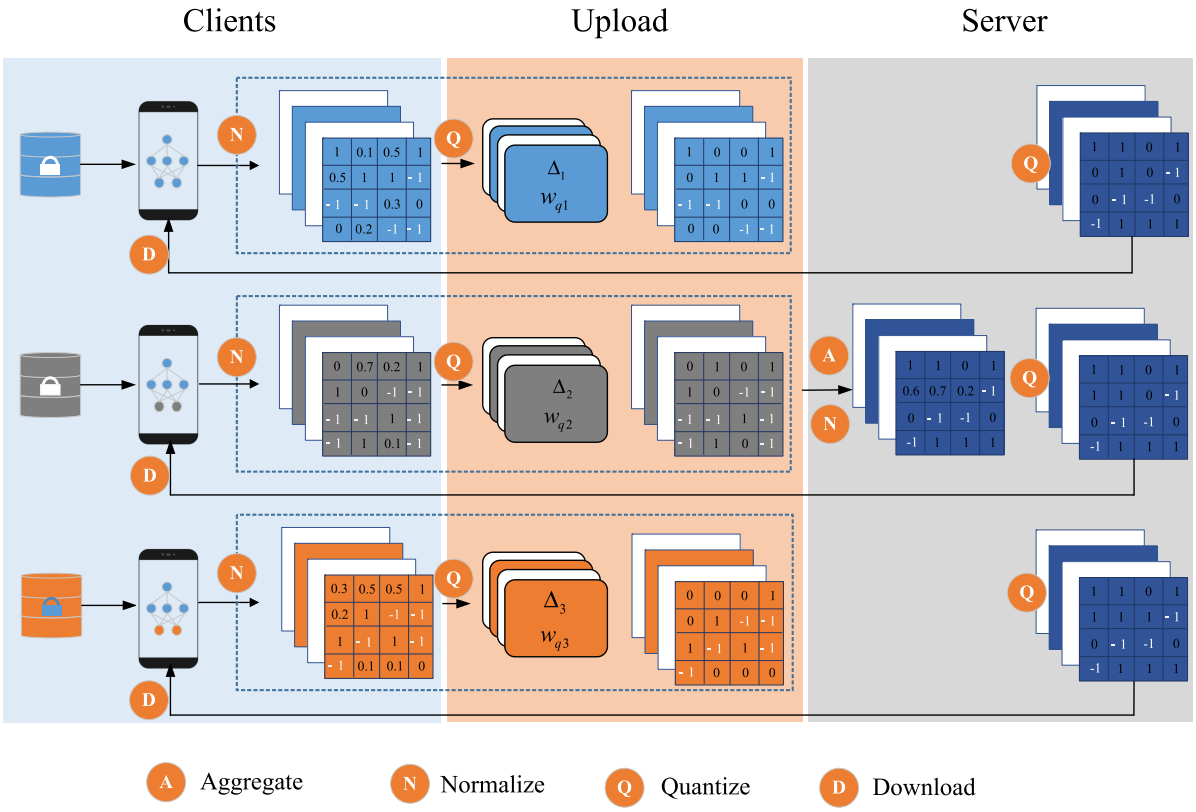

Fig. 4. Diagram of the proposed T-FedAvg. The left part runs on the clients with normalized full-precision weights; then the quantization factors, thresholds, and ternary local models are transmitted to the server, as shown in the middle part; after that, the global model is obtained on the server by aggregation and normalization; finally, the global model is quantized and transmitted to all clients.

where $I_p = \{i|\theta_i \geq \Delta\}$, $I_n = \{j|\theta_j \leq -\Delta\}$, and $I_z = \{k||\theta_k| < \Delta\}$, and according to (4) we have

$$\theta - w_p I_p + w_n I_n = \begin{cases} \theta_i - w_p, & i \in I_p \\ \theta_k, & k \in I_z \\ \theta_j + w_n, & j \in I_n. \end{cases} \tag{23}$$

Then, the original problem can be transformed to

$$\begin{aligned} &\|\theta - w_p I_p + w_n I_n\|_2^2 \\ &= \sum_{i \in I_p} (\theta_i - w_p)^2 + \sum_{j \in I_n} (\theta_j + w_n)^2 + \sum_{k \in I_z} \theta_k^2 \\ &= |I_p| w_p^2 + |I_n| w_n^2 - 2 w_p \sum_{i \in I_p} \theta_i + 2 w_n \sum_{j \in I_n} \theta_j + C \end{aligned} \tag{24}$$

where $C = \sum_{i \in I_p} \theta_i^2 + \sum_{j \in I_n} \theta_j^2 + \sum_{k \in I_z} \theta_k^2$ is a constant independent of w_p and w_n. Hence the optimal solution of (24) can be obtained when

$$\begin{aligned} w_p^* &= \frac{1}{|I_p|} \sum_{i \in I_p} \theta_i \\ w_n^* &= -\frac{1}{|I_n|} \sum_{j \in I_n} \theta_j. \end{aligned} \tag{25}$$

Since the weights are distributed symmetrically, w_p^* and w_n^* will converge to the same value. This completes the proof.

B. Unbiasedness of FTTQ

Here, we first prove the unbiasedness of FTTQ. To simplify the original problem, we make an assumption that is common in network initialization.

With Assumption 1, we prove Proposition 2:

Proposition 2: Let θ be the local scaled network parameters defined in Assumption 1 of one client in a given federated learning system. If θ is quantized by the FTTQ algorithm, then we have

$$\mathbb{E}[\text{FTTQ}(\theta)] = \mathbb{E}(\theta). \tag{26}$$

Proof 2: According to (25), w_q^* is calculated by the elements in $I_p = \{i|\theta_i \geq \Delta\}$, where Δ is a fixed number once the parameters are generated under Assumption 1, hence the elements indexed by I_p obey a new uniform distribution between Δ and 1, then we have

$$\forall i \in I_p, \quad \theta_i \backsim U(\Delta, 1) \tag{27}$$

therefore, the probability density function f of θ_i $(i \in I_p)$ can be regarded as $f(x) = (1/1 - \Delta)$.

According to Proposition 1 and (25), we have

$$\begin{aligned} \mathbb{E}(w^{q*}) &= \mathbb{E}\left(\frac{1}{|I_p|} \sum_{i \in I_p} \theta_i\right) = \frac{1}{|I_p|} \mathbb{E}\left(\sum_{i \in I_p} \theta_i\right) \\ &= \frac{1}{|I_p|} |I_p| \int_{\Delta}^{1} u f(u) du = \int_{\Delta}^{1} u f(u) du \\ &= \frac{1 + \Delta}{2} \end{aligned} \tag{28}$$

Algorithm 2: Ternary Federated Averaging

Input: Initial global model parameters θ_0
Init: Broadcast the global model θ_0 to all clients
for round r = 1,..., T **do**
  for Client $k \in \mathbb{K} = \{1, 2, ..., |\lambda N|\}$ **in parallel do**
    load dataset D_k
    $\theta_k \leftarrow \theta_{r-1}^t$ or θ_{r-1}
    initialize w^q
    $\theta_{k,r}^t \leftarrow$ **FTTQ**(θ_k, w^q)
    upload $\theta_{k,r}^t$ to server
  end
  Server does:
    $\theta_r \leftarrow \sum_{k=1}^{\lambda N} \frac{|D_k|}{\sum_{k=1}^{\lambda N}|D_k|}\theta_{k,r}^t$
    $\Delta^S = 0.05 \times \max(|\theta_r|)$
    $I_p^S,\ I_n^S = \{i \mid \theta_r^i > \Delta^S\},\ \{i \mid \theta_r^i < -\Delta^S\}$
    $w_p^S,\ w_n^S = \frac{1}{|I_p^S|}\sum_i^{|I_p^S|}(|\theta_r^i|),\ \frac{1}{|I_n^S|}\sum_i^{|I_n^S|}(|\theta_r^i|)$
    $\theta_r^t \leftarrow w_p^S \times I_p^S - w_n^S \times I_n^S$
    if θ_r^t *does not crash* **then**
      broadcast θ_r^t /*Strategy I*/
    else
      broadcast θ_r /*Strategy II*/
    end
  end
end

Fig. 5. Ternary federated averaging.

where u is a random number subjected to (27), and $|I_p|$ represents the number of elements in I_p.

We know that

$$\begin{aligned}\mathbb{E}[\text{FTTQ}(\theta)] &= \mathbb{E}\big[w^{q*} \times \text{sign}(\text{mask}(\theta) \times \theta)\big] \\ &= \mathbb{E}(w^{q*})\mathbb{E}\{\text{sign}(\theta)\mathbb{E}[\text{mask}(\theta)]\}\end{aligned} \tag{29}$$

and since

$$\begin{aligned}\mathbb{E}[\text{mask}(\theta)] =\ & P[\text{mask}(\theta) = 1)] \times 1 + P[\text{mask}(\theta) = 0] \\ & \times 0 + P[\text{mask}(\theta) = 1] \times (-1) \\ =\ & \frac{1-\Delta}{2} \times 1 + \Delta \times 0 + \frac{1-\Delta}{2} \times (-1) \\ =\ & 0\end{aligned} \tag{30}$$

hence

$$\begin{aligned}\mathbb{E}[\text{FTTQ}(\theta)] &= \mathbb{E}(w^{q*})\mathbb{E}\{\text{sign}(\theta)\mathbb{E}[\text{mask}(\theta)]\} \\ &= \frac{1+\Delta}{2} \times 0 = 0\end{aligned} \tag{31}$$

and under Assumption 1, we have

$$\mathbb{E}(\theta) = \frac{1 + (-1)}{2} = 0 \tag{32}$$

then it is immediate that

$$\mathbb{E}[\text{FTTQ}(\theta)] = \mathbb{E}(\theta). \tag{33}$$

Hence, the FTTQ quantizer output can be considered as an unbiased estimator of the input [49]. We can guarantee the unbiasedness of FTTQ in federated learning systems when the weights are uniformly distributed.

C. Properties of T-FedAvg

Here, we adopt the following assumption to demonstrate the properties of T-FedAvg, which is widely used in the literature [19].

Assumption 2: When a federated learning system with K clients and one server is established, all clients will be initialized with the same global model.

Furthermore, Zhao *et al.* [19] propose a criterion to define the weight divergence of FedAvg (WD_{Fed}), which is

$$\text{WD}_{\text{Fed}} = ||\theta^{\text{Fed}} - \theta^{\text{Cen}}||/||\theta^{\text{Cen}}||. \tag{34}$$

where θ^{Cen} is the model trained by centralized learning. Here, based on (34), we prove the following proposition:

Proposition 3: Given one layer weight matrix with d dimension in the global model $\theta^{\text{Fed}} \backsim U(-1, 1)$ of a federated learning system, and a quantized model θ^{TFed} using Algorithm 5, then the expected weight divergence of the quantized model will be reduced by $(2\sqrt{2} - \sqrt{7}/2\sqrt{3}||\theta^{\text{Cen}}||)d$, if d^2 is large enough.

Proof 3: To simplify the derivation, we define θ^q as the quantized global model when $\Delta = 0$. According to Section IV-A, we can calculate the weight divergence of quantized federated learning by

$$\text{WD}_{\text{TFed}} = \big\|\theta^{\text{TFed}} - \theta^{\text{Cen}}\big\|/\big\|\theta^{\text{Cen}}\big\|. \tag{35}$$

Then, the numerator of (35) can be rewritten as

$$\big\|\theta^{\text{TFed}} - \theta^{\text{Cen}}\big\| = \big\|w_p^{S*}I_p^S - w_n^{S*}I_n^S - \theta^{\text{Cen}}\big\|. \tag{36}$$

According to Algorithm 5, (25) and Assumption 1, we can calculate the limit of w_p^{S*} and w_n^{S*} when the number of elements in θ^{Cen} is large enough, which are

$$\lim_{d^2 \to +\infty} w_p^{S*} = \lim_{d^2 \to +\infty} w_n^{S*} = \frac{1}{2} \tag{37}$$

where d is the dimension of matrix θ^{Cen} (can also be called the kernel size), and since the size of θ^{Cen} is the same as θ^{TFed}, the index matrices I_p^S and I_n^S can also be used in θ^{Cen}. Then, if (37) holds, the expectation of the squared weight divergence in quantized federated learning equals to

$$\begin{aligned}&\mathbb{E}\Big(\big\|\theta^{\text{TFed}} - \theta^{\text{Cen}}\big\|_2^2\Big)\mathbb{E}\left(\left\|\frac{1}{2}I_p^S - \frac{1}{2}I_n^S - \theta^{\text{Cen}}\right\|_2^2\right) \\ &= \mathbb{E}\left[\sum_{i \in I_p^S}\left(\frac{1}{2} - \theta_i^{\text{Cen}}\right)^2\right] + \mathbb{E}\left[\sum_{j \in I_n^S}\left(\frac{1}{2} + \theta_j^{\text{Cen}}\right)^2\right] \\ &= \frac{d^2}{2}\mathbb{E}\left[\left(\frac{1}{2} - \theta_i^{\text{Cen}}\right)^2\right] + \frac{d^2}{2}\mathbb{E}\left[\left(\frac{1}{2} + \theta_j^{\text{Cen}}\right)^2\right].\end{aligned} \tag{38}$$

Since $\theta_i^{\text{Cen}},\ \theta_j^{\text{Cen}} \backsim U(-1, 1)$, we can define $u_1 \backsim U(-(1/2), (3/2))$, $u_2 \backsim U(-(3/2), (1/2))$ to represent $(1/2) - \theta_i^{\text{Cen}}$ and $(1/2) + \theta_j^{\text{Cen}}$, and the densities of $f(u_1)$,

$f(u_2)$ are equal to $(1/2)$. Hence, (38) can be calculated by

$$\begin{aligned}&\frac{d^2}{2}\mathbb{E}\left[\left(\frac{1}{2}-\theta_i^{\text{Cen}}\right)^2\right]+\frac{d^2}{2}\mathbb{E}\left[\left(\frac{1}{2}+\theta_j^{\text{Cen}}\right)^2\right]\\&=\frac{d^2}{2}\int_{-\frac{1}{2}}^{\frac{3}{2}}u_1^2 f(u_1)du_1+\frac{d^2}{2}\int_{-\frac{3}{2}}^{\frac{1}{2}}u_2^2 f(u_2)du_2\\&=\frac{d^2}{2}\int_{-\frac{1}{2}}^{\frac{3}{2}}\frac{1}{2}u_1^2 du_1+\frac{d^2}{2}\int_{-\frac{3}{2}}^{\frac{1}{2}}\frac{1}{2}u_2^2 du_2=\frac{7}{12}d^2. \quad (39)\end{aligned}$$

Similarly, we can derive the expectation of the numerator of the squared weight divergence of FedAvg by

$$\mathbb{E}\left(||\theta^{Fed}-\theta^{\text{Cen}}||_2^2\right)=\mathbb{E}\left[\sum_{i\in I_p\cup I_n}\left(\theta_i^{Fed}-\theta_i^{\text{Cen}}\right)^2\right]. \quad (40)$$

Consequently, we know that if two independent uniform distributions are on $[-1,\ 1]$, then their sum u_3 has the so-called triangular distribution on $[-2,\ 2]$ with density $f(u_3)=(1/4)(u_3+2)$ when $-2\le u_3\le 0$ and $f(u_3)=(1/4)(-u_3+2)$ when $0\le u_3\le 2$, therefore, we have

$$\begin{aligned}&\mathbb{E}\left[\sum_{i\in I_p^S\cup I_n^S}(\theta_i^{Fed}-\theta_i^{\text{Cen}})^2\right]\\&=\frac{d^2}{2}\int_{-2}^{0}\frac{1}{4}(u_3+2)u_3^2 du_3+\frac{d^2}{2}\int_{0}^{2}\frac{1}{4}(-u_3+2)u_3^2 du_3\\&=\frac{2}{3}d^2. \quad (41)\end{aligned}$$

This completes the proof.

Based on the above proof, we can conclude that quantization can reduce the expected weight divergence in learning from non-IID data in the federated learning framework.

D. Convergence Analysis

If the unbiasedness of FTTQ holds, we have the following proposition that provides the convergence rate of T-FedAvg.

Proposition 4: Let $J_1,\ldots,\ J_N$ be L-smooth and μ-strongly convex, $\mathbb{E}||\nabla J_k(\theta_{k,r},\xi_{k,r})-\nabla J_k(\theta_{k,r})||^2\le\delta_k^2$, $\mathbb{E}||\nabla J_k(\theta_{k,r})||^2\le G^2$, where $\xi_{k,r}$ is a mini-batch sampled uniformly at random from kth client's data. Then, for a federated learning system with N devices (full participation) and IID data distribution, the convergence rate of T-FedAvg is $O(1/NR)$, where R is the total number of SGD iterations performed by each client.

Proof 4: Here we give a brief proof, which heavily relies on the proofs in [50]–[52].

Let

$$\hat{\theta}_R=\sum_{k=1}^{N}\frac{|D_k|}{\sum_{k=1}^{N}|D_k|}\theta_k$$

and correspondingly

$$\hat{\theta}_R^t=\sum_{k=1}^{N}\frac{|D_k|}{\sum_{k=1}^{N}|D_k|}\text{FTTQ}(\theta_k)$$

and according to Proposition 2, we have

$$\begin{aligned}\mathbb{E}(\hat{\theta}_R^t)&=\sum_{k=1}^{N}\frac{|D_k|}{\sum_{k=1}^{N}|D_k|}\mathbb{E}[\text{FTTQ}(\theta_k)]\\&=\sum_{k=1}^{N}\frac{|D_k|}{\sum_{k=1}^{N}|D_k|}\mathbb{E}(\theta_k)\\&=\mathbb{E}(\hat{\theta}_R). \quad (42)\end{aligned}$$

Let $\nu=\max_k N(|D_k|/\sum_{k=1}^{N}|D_k|)$, $\kappa=(L/\mu)$, $\gamma=\max\{32\kappa,E\}$, and set step size $\eta_r=(1/4\mu(\gamma+r))$ according to [52], then based on Proposition 4, we have

$$\mathbb{E}(J(\hat{\theta}_r^t))-J^*=\mathbb{E}(J(\hat{\theta}_r)-J(\theta^*))\le\frac{L}{2}\mathbb{E}||\hat{\theta}_r-\theta^*||^2. \quad (43)$$

Then, according to (42)–(43) and the proof given by Qu *et al.* [52], we can obtain the convergence rate of T-FedAvg by

$$\begin{aligned}&\mathbb{E}(J(\hat{\theta}_r^t))-J^*\\&\le\frac{L}{2}\mathbb{E}||\hat{\theta}_r-\theta^*||^2\\&\le 2\hat{c}||\theta_0-\theta^*||^2\left[\frac{\kappa}{\mu}\frac{1}{N}\frac{1}{R+\gamma}\nu^2\delta^2+96\frac{\kappa^2}{\mu}\frac{1}{(R+\gamma)^2}E^2G^2\right]\\&=O\left(\frac{\kappa}{\mu}\frac{1}{N}\frac{1}{R}\nu^2\delta^2+\frac{\kappa^2}{\mu}\frac{1}{R^2}E^2G^2\right) \quad (44)\end{aligned}$$

where $\hat{c}$ is a large enough constant for inequality scaling. Hence, if we set the local epochs $E=O((R/N)^{1/2})$ then $O(E^2/R^2)=O(1/NR)$, the convergence rates of FedAvg and T-FedAvg will be $O(1/NR)$. The readers are referred to [50]–[52] for detailed proofs.

V. Experimental Results

This section evaluates the performance of the proposed method on widely used benchmark data sets. We set up multiple controlled experiments to examine the performance compared with the standard federated learning algorithm in terms of the test accuracy and communication costs. In the following, we present the experimental settings and the results.

A. Settings

To evaluate the performance of the proposed network quantization and ternary protocol in federated learning systems, we first conduct experiments with five independent physical clients connected by a local area network (LAN).

The physical system consists of one CPU laptop and five GPU workstations that are connected wirelessly through LAN, the CPU laptop acts as the server to aggregate local models and the remaining GPU workstations act as clients participating in the federated training. Each client only communicates with the server and there is no information exchange between the clients.

For simulations, we typically use 100 clients for experiments. A detailed description of the configuration is given below.

1) Compared Algorithms: In this work, we compare the following algorithms:

1) Baseline: the centralized learning algorithm, such as SGD method, which means that all data are stored in a single computing center and the model is trained directly using the entire data.
2) FedAvg: the canonical federated learning approach presented in [16].
   CMFL: a communication-mitigated federated learning algorithm proposed in [35].
3) T-FedAvg: our proposed quantized federated learning approach. Note that the first and last feature layers of the global and local models are full-precision.

2) Data Sets: We select two representative benchmark data sets that are widely used for classification.

1) MNIST [44]: it contains 60 000 training and 10 000 testing gray-scale handwritten image samples with ten classes, where the dimension of each image is 28×28. Since the features of MNIST are easily extracted, this data set is mainly used to train small networks. Since it is relatively simple to extract features from MNIST, no data augmentation method is used in this set of experiments.
2) CIFAR10 [45]: it contains 60 000 colored images of ten types of objects from frogs to planes, 50 000 for training and 10 000 for testing. It is a widely used benchmark data set that is difficult to extract features. Random cropping and horizontal random flipping are chosen for data augmentation. Three color channels are normalized with the mean and standard deviation of $\mu_r = 0.4914$, $\delta_r = 0.247$, $\mu_g = 0.4824$, $\delta_g = 0.244$, and $\mu_b = 0.4467$, $\delta_b = 0.262$, respectively, as recommended in [53].

3) Models: To evaluate the performance of above algorithms, three deep learning models are selected: MLP, CNN, and ResNet*, which represent three popular neural network models. The detailed setting are as follows:

1) MLP is meant for training small data sets, e.g., MNIST. The model contains two hidden layers with the number of neurons of 30 and 20 without bias, ReLU is selected as the activation function. The size of the first and last layer is 784 and 10. Note that the pretrained weights of the first layer do not need to transmitted to the server, which can reduce the order of magnitude of the communication volume, since the number of neurons in this layer is much more than other layers.
2) CNN is a shallow convolutional neural network, which consists of five convolutional layers and three fully connected layers. The first convolutional layer uses the ReLU function, while the remaining ones are followed by a batch normalization layer, a ReLU function, and a max pooling layer.
3) ResNet18* is a simplified version of the widely used ResNet [54], where the number of input and output channels for all convolutional layers is reduced to 64. It is a typical benchmark model for evaluating the performance of algorithms on large data sets.

4) Data Distribution: The performance of federated learning is affected by the features of training data stored on the separated clients. To investigate the impact of different data distributions, several types of data are generated:

1) IID data: each client holds an IID subset of data containing ten classes, thus having an IID-subset of the data.
2) Non-IID data: the union of the samples in all clients is the entire data set, but the number of classes contained in each client is not equal to the total number of categories in the entire data set (ten for MNIST and CIFAR10). We can use the label to assign samples of N_c classes to each client, where N_c is the number of classes per client. In case of extremely non-IID data, N_c is equal to 1 for each client, but this case is generally not considered since there is no need to train (e.g., classification) if only one class is stored on each client.
3) Unbalancedness in data size: typically, the size of the data sets on different clients varies a lot. To investigate the influence of the unbalancedness in data size in the federated learning environment, we split the entire data set into several distinct parts.

5) Basic Configuration: The basic configuration of the federated learning system in our experiments is set as follows:

1) Total number of clients: $N = 100$ for MNIST with simulation, and $N = 5$ for CIFAR10 (as suggested in [53]) using the physical system.
2) The participation ratio per round: $\lambda = 0.1$ for MNIST and $\lambda = 1$ for CIFAR10.
3) Classes per client: $N_c = 10$.
4) Local epochs: $E = 5$ for MNIST, and $E = 10$ for CIFAR10.

The batch size is fixed to 64 and the same learning rate is used for both the centralized and federated algorithms with an exponential decay (with a rate is 0.95) for every five communication rounds in all CIFAR10 experiments, as suggested in [50]. The number of samples per client is equal to the size of the data set divided by N.

For a fair comparison, we assess the performance of each compared algorithm after 100 iterations of training with the same amount of training samples, which is averaged over five independent runs. Take MNIST as an example, there are 100 clients, each client has 600 samples, and 100 rounds are run for federated learning. Correspondingly, 60 000 samples are used to train the centralized methods for 100 epochs. It should be pointed out that since the participating ratio λ is less than or equal to 1, the number of used training samples in federated learning is always smaller than that used in centralized learning. The hyperparameters and test results obtained by the baseline models are summarized in Table I.

TABLE I
MODELS AND HYPERPARAMETERS

Models	MLP	CNN	ResNet*
Dataset	MNIST	CIFAR10	CIFAR10
Optimizer	SGD	Adam	Adam
Learning rate	0.01	0.001	0.008
Basic accuracy(%)	92.25 $\pm$ 0.06	85.63 $\pm$ 0.10	88.80 $\pm$ 0.20

TABLE II

COMMUNICATION COSTS IN 100 ROUNDS ON IID DATA. FOR MLP, 10 OUT OF 100 CLIENTS ($\lambda = 0.1$) PARTICIPATE IN EACH ROUND OF TRAINING. FOR CIFAR10, ALL FIVE CLIENTS PARTICIPATE IN EACH ROUND OF TRAINING ($\lambda = 1.0$)

Methods	MLP	CNN	ResNet*
	upload / download (MB)	upload / download (MB)	upload / download (MB)
FedAvg	19.53 / 19.53	16196.59 / 16196.59	9253.81/ 9253.81
CMFL	5.63 / 19.53	4079.75 / 16196.59	2666.80 / 9253.81
T-FedAvg	2.36 / 2.36	2201.84 / 3041.53	1229.27 / 6847.82

B. Results on IID Data

In this part, we conduct experiments comparing the communication costs and performance of MLP, CNN, and ResNet* on IID MNIST and CIFAR10 using the three federated learning frameworks and centralized learning (baseline). Specifically, for the MNIST data set, there are 100 clients, each having 600 samples with IID distribution, and λ is set to 0.1. For CIFAR10, 10000 IID training samples are stored on each client, and there are five clients with $\lambda = 1$.

First of all, we compare the communication costs of FedAvg, CMFL, and T-FedAvg with a fixed number of rounds. For T-FedAvg, we calculate the upstream communication costs by treating the ternary parts of local models as 2-bit, and we calculate the downstream cost approximately by

$$\text{download} = \left(1 - \frac{S_I}{T}\right) \times \text{upload}_f + \frac{S_I}{T} \times \text{upload}_t \quad (45)$$

where T is the number of total rounds in Algorithm 5, and S_I is the number of times the server implements Strategy I. The results are listed in Table II, where upload_f and upload_t are the upstream communication costs of FedAvg and T-FedAvg, respectively.

It is found that Strategy II is not used by the server when training MLP, hence the communication costs of T-FedAvg are reduced by 88% in the upload and download phases compared to the standard FedAvg. When training CNN, the uploaded communication costs of T-FedAvg are reduced to 13%, and the download communication costs are 82% of FedAvg and CMFL, which means that the server implements Strategy II for several times to maintain the performance. Similarly, T-FedAvg compresses nearly 94% of the upstream, 25% of the downstream communications, and outperforms CMFL when training ResNet*. Note that we have not transferred the parameters of the first layer in MLP to the server.

The detailed test accuracies are given in Table III. From the table, we see that MLP using centralized learning achieves an accuracy of 92.25%. By contrast, MLP using CMFL and T-FedAvg achieve 90.91% and 91.95%, respectively, both outperforming FedAvg although slightly underperformed by the baseline mode. This may be attributed to the fact that much less data are involved in training in the federated learning frameworks than in the baseline, since the participation ratio is set to 0.1. It is interesting that the quantized T-FedAvg outperforms FedAvg, and one possible reason is that the ternary layer may play the role of weight regularization in MLP.

TABLE III

TEST ACCURACIES ACHIEVED AND WEIGHTS WIDTH OF DIFFERENT ALGORITHMS WHEN TRAINED ON IID DATA

Accuracy(%)	MNIST	CIFAR10		Width
	MLP	CNN	ResNet*	bit
Baseline	92.25 ± 0.06	85.63 ± 0.10	88.80 ± 0.20	32
FedAvg	90.63 ± 0.17	85.47 ± 0.14	88.34 ± 0.40	32
CMFL	90.91 ± 0.36	84.23 ± 0.71	87.13 ± 0.66	32
T-FedAvg	91.95 ± 0.11	84.46 ± 0.17	87.87 ± 0.18	2

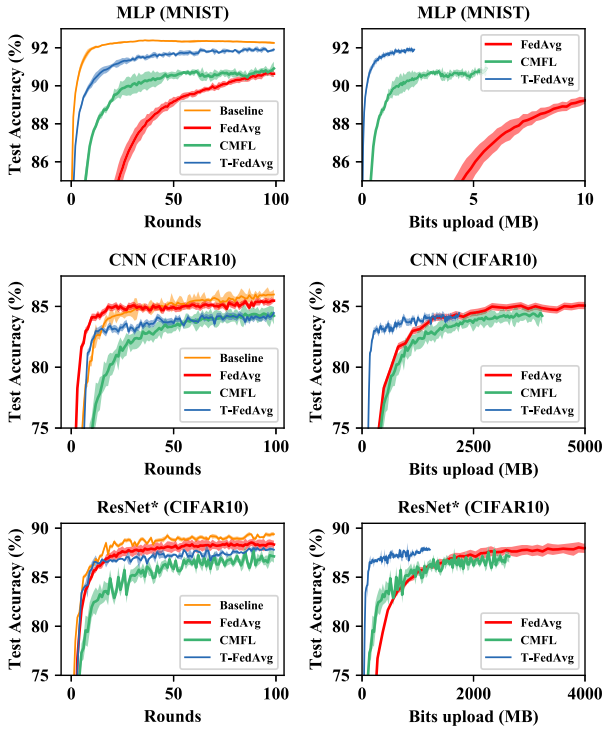

Fig. 6. Convergence curves over rounds and bits upload of the compared algorithms. For each algorithm, the solid line denotes the averaged test accuracy over five independent runs, and the shaded region denotes the standard deviation.

T-FedAvg using CNN achieves 84.46% while FedAvg achieves 85.47%, and they achieve 87.87% and 88.34% test accuracies on CIFAR10 with ResNet*, respectively. The test accuracy of CMFL on CIFAR10 is not as good as expected, which may be affected by the reduction in communication costs. Generally speaking, T-FedAvg underperforms FedAvg under a limited weight width, which is known as the quantization error. However, the model size of the ternary part in the global and local models of T-FedAvg is only 1/16 compared to the full-precision models, and CMFL reduces a large number of communication rounds. Thus, we consider that both methods have achieved satisfying performance in the federated learning framework.

To take a closer look at the performance of the four algorithms under comparison, we also plot the convergence curves in Fig. 6. From these results, we find that T-FedAvg

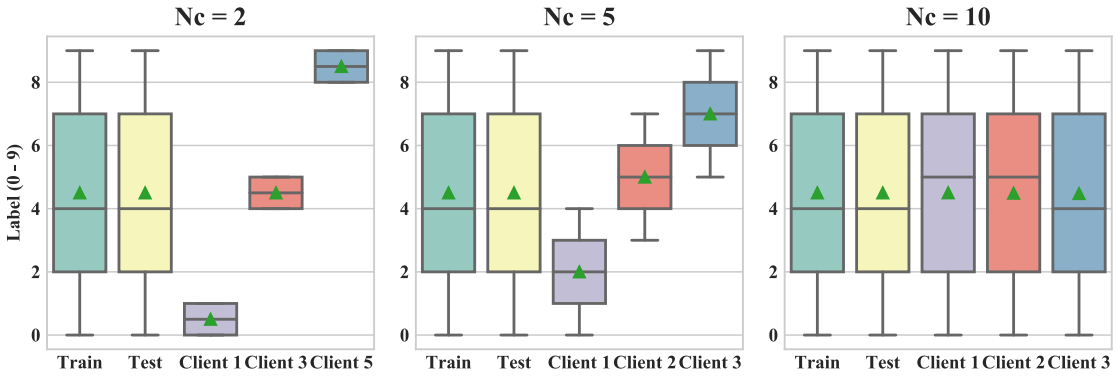

Fig. 7. Data distributions with different N_c. When $N_c = 2$, there is no overlap in data between clients and each client contains two categories (left). When $N_c = 5$, the samples on ten clients are sampled by labels but there are some overlap between clients (middle). When $N_c = 10$, the samples on ten clients are generated by randomly sampling (right). Note that only three clients are shown in the figures when the N_c is 2, 5, or 10.

converges faster than CMFL in all test instances, although FedAvg converges slightly faster than T-FedAvg when CNN is trained on CIFAR10. However, it is clear to see that T-FedAvg converges the fastest over the bits upload among the three federated learning frameworks in the three instances. This can be attributed to the fact that FTTQ optimizes the quantized network based on the full-precision model during the training process. Finally, it is noted that the federated learning methods converge more slowly than the centralized learning algorithm.

C. Performance Analysis of Non-IID Data

Since the communication costs of the three algorithms under comparison on non-IID data are exactly the same as on IID data (refer to Table II), here we focus on the influence of the non-IID data for classification tasks.

In federated learning, the data distribution is considered to be non-IID when N_c is smaller than the number of total classes in the training data. The data distributions used in the following experiments are depicted in Fig. 7, where the y-axis represents the sample label (0–9). As shown in the right panel of the figure, the original distributions of training and test data are IID. That is, when N_c equals 10, where each client has an IID subset of the entire data set. When the $N_c = 2$, the samples on each client are divided according to the label, which is non-IID. Similarly, the samples in all clients are non-IID when N_c is equal to 5, but there are some overlaps in data between the clients.

The experimental results are presented in Table IV. These results indicate that T-FedAvg performs either comparably well or even better than FedAvg and CMFL on both MNIST and CIFAR10. In particular, T-FedAvg has achieved 5.22% and 6.43% performance enhancement, respectively, compared with FedAvg and CMFL on CIFAR10 when $N_c = 5$. In the following, we provide some empirical analysis of why quantization might enhance the learning performance on non-IID data.

Fig. 8 plots the convergence curves of the test performance of the three compared algorithms over communication rounds and bits uploads, from which we see that both T-FedAvg and FedAvg converge better than CMFL over the training rounds, and T-FedAvg becomes better than FedAvg at the later stage.

TABLE IV
TEST ACCURACIES ACHIEVED OVER NON-IID DATA FOR DIFFERENT N_c

Accuracy (%)	MNIST		CIFAR10 (ResNet*)	
	$N_c = 2$	$N_c = 5$	$N_c = 2$	$N_c = 5$
FedAvg	82.61 ± 4.31	89.24 ± 0.38	40.17 ± 5.6	71.47 ± 0.23
CMFL	81.51 ± 2.29	88.30 ± 0.52	39.30 ± 6.1	70.26 ± 0.63
T-FedAvg	87.29 ± 0.89	90.04 ± 0.68	40.46 ± 4.3	76.69 ± 0.67

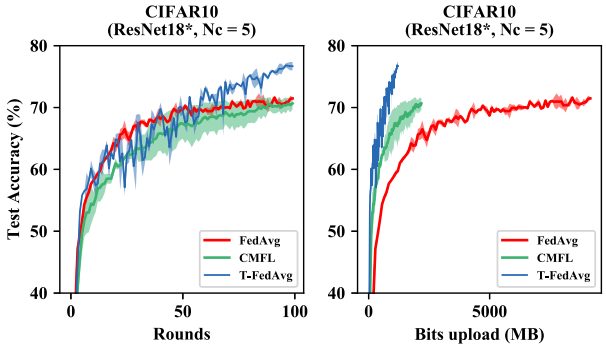

Fig. 8. Convergence trends and communication costs of FedAvg, CMFL, and T-FedAvg when $N_c = 5$.

Meanwhile, it is clear that T-FedAvg converges the fastest and the best over bits upload, confirming the performance we see in Table IV.

The better performance of T-FedAvg observed on the CIFAR10 data set appears counter-intuitive. However, as it is proved in Section IV-C that if Assumption 1 holds, quantization is able to reduce weight divergence on non-IID data in federated learning, thereby achieving more robust learning. Our results shown in Fig. 8 also confirm this performance enhancement of T-FedAvg compared to FedAvg.

It should be pointed out, however, that performance drop in federated learning on non-IID data compared to on IID data remains an open challenge [16], since the local stochastic gradients cannot be considered as an unbiased estimate of the global gradients [19] when the data are non-IID. Theoretically, since T-FedAvg could reduce the upstream and downstream

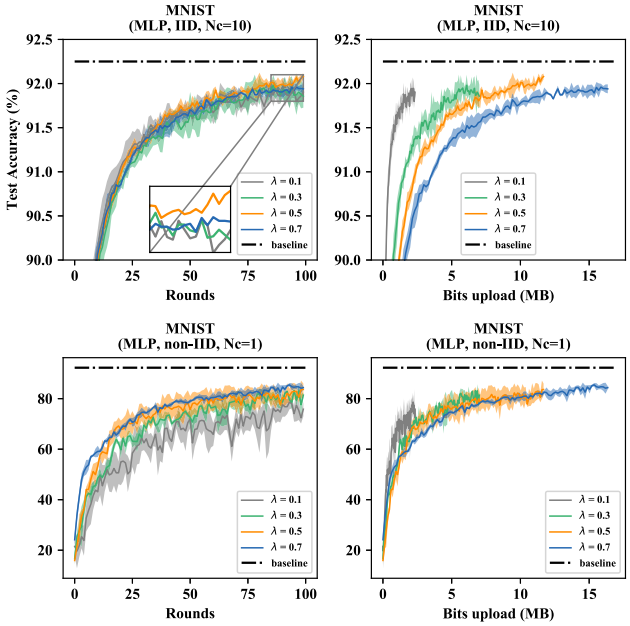

Fig. 9. Test accuracies achieved by T-FedAvg and uploaded bits when training MLP on MNIST with IID and non-IID distribution in fixed rounds at different participation ratios (0.1, 0.3, 0.5, 0.7).

communication costs, we can increase the number of communication rounds within the same constraint of budgets to alleviate the performance degeneration.

D. Influence of the Participation Ratio

We investigate the effect of λ on T-FedAvg in this subsection. We fix the total number of the clients and the local batch sizes to 100 and 64, respectively, throughout all experiments. Here, the experiments are done using MLP only, since the robustness of MLP to non-IID data (see in Table IV) can reduce the effect of the model used. Fig. 9 presents the test accuracies achieved by T-FedAvg during the training on IID and non-IID MNIST in the federated learning environment with different participation ratios (λ), where we also give the comparison of the uploaded bits under different λ.

As we can see, T-FedAvg is relatively robust to the changes of the participation ratio λ over IID and non-IID data, and the performance fluctuations decrease as λ increases. Generally speaking, reducing participation ratios λ usually has negative effects on the learning speed and the final accuracy achieved in a fixed number of rounds, the negative effects are more pronounced on non-IID data (refer to the right panel of Fig. 9). However, once λ is increased to a certain value, the performance improvement will be less significant, and sometimes it even starts to degrade. We surmise that the performance degradation on non-IID data is heavily dependent on the representativeness of the selected local models.

Intuitively, the upstream and downstream communication costs increase as λ increases. Correspondingly, the more clients are involved, the more communication costs will be saved compared to vanilla FedAvg. This is encouraging since the number of clients is usually very large in real-world applications.

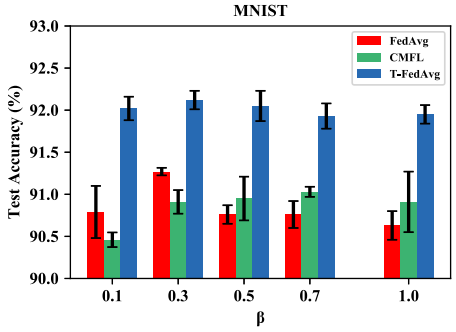

Fig. 10. Test accuracies achieved by MLP on MNIST after 100 rounds of iterations with FedAvg, CMFL, and T-FedAvg, where the number of clients and participation ratio are set to 100 and 0.1, respectively.

E. Influence of Unbalancedness in Data Size

All experiments above were performed with a balanced split of the data, where all clients were assigned the same number of samples. In the following, we investigate the performance of the proposed algorithm on the unbalancedness in the data size [43]. If we use $S_N = \{|D_1|, |D_2|, \ldots, |D_N|\}$ to represent the set of number of samples on N clients, we can define the degree of unbalanceness by the ratio β:

$$\beta = \frac{\text{median}\{S_N\}}{\max\{S_N\}} \tag{46}$$

where the median of S_N is sometimes helpful to accommodate long tailed distributions and possible outliers [55].

When $\beta = 0.1$, most of the samples are stored on a few clients, and when $\beta = 1$, almost all clients store the same number of samples. To simulate the unbalanced data distribution, we vary β from 0.1 to 1, with an average of 30 out of 100 clients being participating. And the test accuracies achieved by FedAvg and T-FedAvg for various β are illustrated in Fig. 10.

We can see that the unbalancedness does not have a significant impact on the performance of the federated learning algorithms. This is due to the fact that the local models are able to learn properly with IID data, even when the data are unevenly distributed on the clients.

F. Comparison of Time Complexity

In this section, we first analyze the computational complexity of FedAvg and T-FedAvg, then conduct empirical experiments to compare the runtime of different federated learning algorithms.

According to [56], for one mini-batch training on each client, the time complexity of FedAvg is $O(BD)$, where B is the batch size and D is the number of elements in θ, assuming there is only one layer in the network. Correspondingly, the time complexity of T-FedAvg in each iteration can be divided into three parts: $O((D/2) + 1)$ for calculating the derivative of w^q by (18), $O(BD)$ for the derivative of θ by (19), and $O(D)$ for quantization. Hence, the overall time complexity of T-FedAvg in each iteration on each client is $O((B + (3/2))D)$. As for the server, the averaging time complexity of FedAvg and T-FedAvg is $O((\lambda N - 1)D)$ and

TABLE V
ELAPSED TIME FOR EACH CLIENT TO COMPLETE FIVE LOCAL EPOCH OVER DIFFERENT FEDERATED LEARNING ALGORITHMS

Time (s)	MNIST	CIFAR10	
	MLP	CNN	ResNet*
FedAvg	1.06 ± 0.03	7.92 ± 0.34	21.74 ± 0.38
CMFL	1.62 ± 0.06	12.48 ± 0.18	27.68 ± 0.35
T-FedAvg	1.06 ± 0.10	8.97 ± 0.23	38.22 ± 0.50

$O(\lambda ND)$, respectively, where λ is the participation ratio, N is the total number of clients and $\lambda N \geq 1$.

Now we investigate the runtime of different federated learning algorithms. The PyTorch framework [57] is used to implement three methods on a workstation with Intel Xeon(R) CPU E5-2620 v4 at 2.10GHz × 32 and 1 NVIDIA GTX1080Ti GPU, and the system version is Ubuntu 16.04 LTS. The runtime of each client for five local epochs is counted, the results obtained in 20 independent runs are presented in Table V.

As we can see, when training MLP, the elapsed time of T-FedAvg is almost the same as FedAvg and is much less than CMFL. However, as the depth of global model increases, the time complexity of our algorithm will increase significantly. Indeed, this will be a potential limitation of model quantization when the model becomes deeper, despite that the proposed method has already reduced a large number of quantization factors. In the future, we will work out solutions to reduce the time complexity of T-FedAvg, for example, by implementing quantization only on the last local epoch.

VI. CONCLUSION AND FUTURE WORK

Federated learning is effective in privacy preservation, although it is constrained by limited upstream and downstream bandwidths and the performance may seriously degrade when the data distribution is non-IID. To address these issues, we have proposed FTTQ, a compression method adjusted for federated learning based on TTQ algorithm, to reduce the energy consumption at the inference stage on the clients. Furthermore, we have proposed ternary federated learning protocol, which compress both uploading and downloading communications. Detailed theoretic proofs of the unbiasedness of quantization, and reduction of weight divergence. Our experimental results on widely used benchmark data sets demonstrate the effectiveness of the proposed algorithms. Moreover, since we have reduced the downstream and upstream communication costs between the clients and server, we can increase the number of clients or the rounds of communications within the same constraint of budgets to improve the performance of federated learning.

Our approach can be seen as an application of trained ternary quantization method by quantizing the local and global model to reduce the communication costs. However, the reduction in communication costs is at the expense of increased computational costs on the clients, in particular when the model becomes very deep. Our future work will aim at finding more efficient approaches to improving the performance of federated learning on non-IID data and more efficient encoding strategies for quantization.

REFERENCES

[1] G. E. Hinton, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[3] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, p. 484, 2016.

[4] P. Voigt and A. Von dem Bussche, "The eu general data protection regulation (GDPR)," in *A Practical Guide*, 1st ed. Cham, Switzerland: Springer, 2017.

[5] J. Dean *et al.*, "Large scale distributed deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1223–1231.

[6] N. D. Vanli, M. O. Sayin, I. Delibalta, and S. S. Kozat, "Sequential nonlinear learning for distributed multiagent systems via extreme learning machines," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 3, pp. 546–558, Mar. 2017.

[7] M. Duan, K. Li, X. Liao, and K. Li, "A parallel multiclassification algorithm for big data using an extreme learning machine," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2337–2351, Jun. 2018.

[8] 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.

[9] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein, "Distributed graphLab: A framework for machine learning in the cloud," *Proc. VLDB Endowment.*, vol. 5, no. 8, pp. 716–727, 2012.

[10] A. Smola and S. Narayanamurthy, "An architecture for parallel topic models," *Proc. VLDB Endowment*, vol. 3, nos. 1–2, pp. 703–710, Sep. 2010.

[11] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous distributed systems," 2016, *arXiv:1603.04467*. [Online]. Available: http://arxiv.org/abs/1603.04467

[12] M. Li, "Scaling distributed machine learning with the parameter server," in *Proc. Int. Conf. Big Data Sci. Comput. BigDataScience*, 2014, pp. 583–598.

[13] J. Zhang and O. Simeone, "LAGC: Lazily aggregated gradient coding for straggler-tolerant and communication-efficient distributed learning," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Apr. 3, 2020, doi: 10.1109/TNNLS.2020.2979762.

[14] S. Zhang, C. Zhang, Z. You, R. Zheng, and B. Xu, "Asynchronous stochastic gradient descent for DNN training," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 6660–6663.

[15] K. Hsieh *et al.*, "Gaia: Geo-distributed machine learning approaching LAN speeds," in *Proc. USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2017, pp. 629–647.

[16] H. B. McMahan *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, 2016, pp. 1273–1282.

[17] J. Konečný, B. McMahan, and D. Ramage, "Federated optimization:Distributed optimization beyond the datacenter," 2015, *arXiv:1511.03575*. [Online]. Available: http://arxiv.org/abs/1511.03575

[18] J. Konečný, H. Brendan McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," 2016, *arXiv:1610.02527*. [Online]. Available: http://arxiv.org/abs/1610.02527

[19] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-IID data," 2018, *arXiv:1806.00582*. [Online]. Available: http://arxiv.org/abs/1806.00582

[20] S. Wang *et al.*, "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, Jun. 2019.

[21] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, p. 12, 2019.

[22] 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.

[23] K. Bonawitz *et al.*, "Towards federated learning at scale: System design," 2019, *arXiv:1902.01046*. [Online]. Available: http://arxiv.org/abs/1902.01046

[24] 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.

[25] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.

[26] speedtest.net. (2016). *United Kingdom Mobile Speedtest Data*. [Online]. Available: https://www.speedtest.net/reports/united-kingdom/

[27] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Proc. Adv. Neural Inf. Process. Syst.*, 1990, pp. 598–605.

[28] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1135–1143.

[29] R. Rigamonti, A. Sironi, V. Lepetit, and P. Fua, "Learning separable filters," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 2754–2761.

[30] T. Cohen and M. Welling, "Group equivariant convolutional networks," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2990–2999.

[31] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*. [Online]. Available: http://arxiv.org/abs/1503.02531

[32] N. Passalis and A. Tefas, "Unsupervised knowledge transfer using similarity embeddings," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 3, pp. 946–950, Mar. 2019.

[33] T. Guo, C. Xu, S. He, B. Shi, C. Xu, and D. Tao, "Robust student network learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 7, pp. 2455–2468, Jul. 2020.

[34] 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.

[35] L. Wang, W. Wang, and B. Li, "CMFL: Mitigating communication overhead for federated learning," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2019, pp. 954–964.

[36] 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.

[37] W. Wen *et al.*, "TernGrad: Ternary gradients to reduce communication in distributed deep learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1509–1519.

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

[39] M. Courbariaux, Y. Bengio, and J. P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Proc. Conf. Neural Inf. Process. Syst. (NIPS)*, 2015, pp. 3123–3131.

[40] F. Li, B. Zhang, and B. Liu, "Ternary weight networks," in *Proc. Conf. Neural Inf. Process. Syst. (NIPS) Workshop*, 2016.

[41] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1709–1720.

[42] J. Chen, L. Liu, Y. Liu, and X. Zeng, "A learning framework for n-Bit quantized neural networks toward FPGAs," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Apr. 3, 2020, doi: 10.1109/TNNLS.2020.2980041.

[43] 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.

[44] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[45] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, Tech. Rep. TR-2019, 2009.

[46] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "DoReFa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," 2016, *arXiv:1606.06160*. [Online]. Available: http://arxiv.org/abs/1606.06160

[47] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," in *Proc. Int. Conf. Learn. Represent.*, 2017.

[48] A. Polino, R. Pascanu, and D. Alistarh, "Model compression via distillation and quantization," 2018, *arXiv:1802.05668*. [Online]. Available: http://arxiv.org/abs/1802.05668

[49] R. M. Gray and D. L. Neuhoff, "Quantization," *IEEE Trans. Inf. Theory*, vol. 44, no. 6, pp. 2325–2383, Oct. 1998.

[50] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-IID data," in *Proc. Int. Conf. Learn. Represent.*, 2020.

[51] S. U. Stich, "Local SGD converges fast and communicates little," in *Proc. Int. Conf. Learn. Represent.*, 2019.

[52] Z. Qu, K. Lin, J. Kalagnanam, Z. Li, J. Zhou, and Z. Zhou, "Federated Learning's blessing: FedAvg has linear speedup," 2020, *arXiv:2007.05690*. [Online]. Available: http://arxiv.org/abs/2007.05690

[53] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, "Federated learning with matched averaging," in *Proc. Int. Conf. Learn. Represent.*, 2020.

[54] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[55] A. W. Bowman and A. Azzalini, *Applied Smoothing Techniques for Data Analysis: The Kernel Approach With S-Plus Illustrations*, vol. 18. Oxford, U.K.: Oxford Univ. Press, 1997.

[56] Y. Mu, W. Liu, X. Liu, and W. Fan, "Stochastic gradient made stable: A manifold propagation approach for large-scale optimization," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 2, pp. 458–471, Feb. 2017.

[57] A. Paszke *et al.*, "Automatic differentiation in pytorch," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS) Workshop*, 2017.

Jinjin Xu received the B.E. degree from the School of Information Science and Technology, East China University of Science and Technology, Shanghai, China, in 2017, where he is currently pursuing the Ph.D. degree.

His current research interests include federated learning, data-driven optimization, and their applications.

Wenli Du received the B.S and M.S. degrees in chemical process control from the Dalian University of Technology, Dalian, China, in 1997 and 2000, respectively, and the Ph.D. degree in control theory and control engineering from the East China University of Science and Technology, Shanghai, China, in 2005.

She is currently a Professor and the Dean of the College of Information Science and the Vice Dean of the Key Laboratory of Advanced Control and Optimization for Chemical Process, Ministry of Education, East China University of Science and Technology. Her research interests include control theory and applications, system modeling, advanced control, and process optimization.

Yaochu Jin (Fellow, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees from Zhejiang University, Hangzhou, China, in 1988, 1991, and 1996, respectively, and the Dr.-Ing. degree from Ruhr University, Bochum, Germany, in 2001.

He was a Finland Distinguished Professor with the University of Jyväskylä, Jyväskylä, Finland, and Changjiang Distinguished Visiting Professor with the Northeastern University, Shenyang, China. He is currently a Distinguished Chair Professor in Computational Intelligence with the Department of Computer Science, University of Surrey, Guildford, U.K., where he heads the Nature Inspired Computing and Engineering Group. His main research interests include evolutionary computation, multiobjective machine learning, secure machine learning, and self-organizing collective systems.

Dr. Jin was a recipient of the 2014, 2016, and 2019 *IEEE Computational Intelligence Magazine* Outstanding Paper Award, the 2018 and 2020 IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION Outstanding Paper Award, and the Best Paper Award of the 2010 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology. He was named 2019–2020 Highly Cited Researchers by the Web of Science Group. He is presently the Editor-in-Chief of the IEEE TRANSACTIONS ON COGNITIVE AND DEVELOPMENTAL SYSTEMS and *Complex & Intelligent Systems*. He was an IEEE Distinguished Lecturer for the period from 2013 to 2015 and 2017 to 2019.

Wangli He (Senior Member, IEEE) received the B.S. degree in information and computing science and the Ph.D. degree in applied mathematics from Southeast University, Nanjing, China, in 2005 and 2010, respectively.

She was a Post-Doctoral Research Fellow with the School of Information Science and Engineering, East China University of Science and Technology, Shanghai, China, and a Visiting Post-Doctoral Research Fellow with Central Queensland University, Rockhampton, QLD, Australia, from 2010 to 2011. During 2013 to 2017, she held several visiting positions with University of Hong Kong, Hong Kong, City University of Hong Kong, Hong Kong, Potsdam Institute for Climate Research Institute, Potsdam, Germany, and Tokyo Metropolitan University, Hachioji, Japan. She is currently a Professor with the School of Information Science and Engineering, East China University of Science and Technology. Her current research interests include cooperative control and distributed optimization of multiagent systems, networked nonlinear systems, and security of cyber-physical systems.

She was the Chair of Technical Committee on Networked-based Control Systems and Applications of IES (2018–2019) and Visiting Associate Professor of Tokyo Metropolitan University (2015–2017). She was the recipient of National Outstanding Youth Science Foundation in 2019 and The Sixth Young Scientist Award of Chinese Association of Automation in 2020. She won the first prize of Shanghai Natural Science Award in 2019. She is an Associate Editor of several international journals including IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS and the IEEE JOURNAL OF EMERGING AND SELECTED TOPICS IN INDUSTRIAL ELECTRONICS.

Ran Cheng (Member, IEEE) received the B.Sc. degree from the Northeastern University, Shenyang, China, in 2010, and the Ph.D. degree from the University of Surrey, Guildford, U.K., in 2016.

He is currently an Associate Professor with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China.

Dr. Cheng was a recipient of the 2018 IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION Outstanding Paper Award, the 2019 IEEE Computational Intelligence Society (CIS) Outstanding Ph.D. Dissertation Award, and the 2020 *IEEE Computational Intelligence Magazine* Outstanding Paper Award. He is the founding Chair of IEEE Symposium on Model Based Evolutionary Algorithms (IEEE MBEA). He is an Associate Editor of *IEEE Transactions on Artificial Intelligence*.