Ternary Compression for Communication-Efficient Federated Learning

Journal:	*IEEE Transactions on Neural Networks and Learning Systems*
Manuscript ID	TNNLS-2020-P-13221
Manuscript Type:	Regular Paper
Date Submitted by the Author:	28-Jan-2020
Complete List of Authors:	Xu, Jinjin; East China University of Science and Technology, Laboratory of Advanced Control and Optimization for Chemical Processes Du, Wenli; East China University of Science and Technology, Key Laboratory of Advanced Control and Optimization for Chemical Processes Cheng, Ran; Southern University of Science and Technology, Department of Computer Science and Engineering He, Wangli; East China University of Science and Technology, School of Information Science and Engineering Jin, Yaochu; University of Surrey, Department of Computing;
Keywords:	Deep learning, federated learning, communication efficiency, ternary coding

Ternary Compression for Communication-Efficient Federated Learning

Jinjin Xu, Wenli Du, Ran Cheng, Wangli He, *Senior member, IEEE*, and Yaochu Jin, *Fellow, 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 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. A convergence proof of the quantization factor and the unbiasedness of FTTQ is given. In addition, 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 datasets, and our results demonstrate the effectiveness of FTTQ and T-FedAvg compared with the canonical federated learning algorithms in reducing communication costs and maintaining the learning performance.**

***Index Terms*—Deep learning, federated learning, communication efficiency, ternary coding.**

I. Introduction

THE number of Internet of Things (IoTs) and smart mobile devices deployed, e.g., in process industry has grown dramatically over the past decades, having generated massive amounts of data stored distributively every moment. Meanwhile, recent achievements in deep learning [1], such as AlphaGo [2], rely heavily on the knowledge stored in big data. Naturally, to adopt deep learning methods for effective utilization of the rich data contained in local clients of the process industry, e.g., branch factories, will provide a strong support to industrial production. However, training deep learning models by distributed data is difficult, while uploading private data to cloud is controversial, due to limitations on network bandwidth, budgets and security regulations, e.g., GDPR [3].

Many research efforts have been devoted to related fields recently, while early work in this area has mainly focused on training deep models on multiple machines to alleviate the computational burden of large data volumes, known as distributed machine learning [4]. These methods have achieved satisfactory performance, by splitting big data into tiny sets to accelerate the model training process. For example, data parallelism [4], model parallelism [4], [5] (see in Fig. 1) and parameter server [6]–[8] are commonly used methods in practice. Correspondingly, the weights optimization strategies for multiple machines have also been proposed. For example, Zhang et al. [9] proposed asynchronous mini-batch stochastic gradient descent algorithm (ASGD) on multi-GPU devices for deep neural networks (DNNs) training and achieved 3.2 times speed-up on 4 GPUs than the single one without loss of precision. Recently, distributed machine learning algorithms for multiple datacenters which are located in different regions have been studied in [10]. However, little attention has been paid to the data security and the impact of data distribution on the performance.

To address the drawbacks of distributed learning, researchers have proposed an interesting framework to train a global model while keeping the private data locally, known as federated learning [11]–[13]. The federated approach makes it possible to extract knowledge in the data distributed on local devices without uploading private data to a certain server. Fig. 2 illustrates the simplified workflow and an application scenario in the process industry. Several extensions have been introduced to the standard federated learning system. Zhao et al. [14] have observed weight divergence caused by extreme data distributions and proposed a method of sharing a small amount of data with other clients to enhance the performance of federated learning algorithms. Wang et al. [15] have proposed adaptive federated learning systems under a given resource 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 leanring can be found in [16], [17], and design ideas, the challenges and future research

Manuscript received xx, 2020; revised xxx, 2020. This work was supported by National Natural Science Foundation of China (Basic Science Center Program: 61988101), International (Regional) Cooperation and Exchange Project (1720106008), National Natural Science Foundation of China (Major Program: 61590923), National Natural Science Fund for Distinguished Young Scholars (61725301), National Natural Science Foundation of China (Major Program: 61590922) and China Scholarship Council (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 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

Ran Cheng is the Shenzhen Key Laboratory of Computational Intelligence, University Key Laboratory of Evolving Intelligent Systems of Guangdong Province, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China. Email: chengr@sustech.edu.cn.

Yaochu Jin is with the Department of Computer Science, University of Surrey, Guildford, GU27XH, UK. E-mail:yaochu.jin@surrey.ac.uk.

directions of federated learning on massively mobile devices are presented in [18], [19].

Since users may pay more attention to privacy protection and data security, federated learning will play a key role in deep learning, although it faces challenges in terms of data distribution and communication costs.

1) Data Distribution: The data generated by different clients, e.g., factories, may be unbalanced and not subject to the independent and identical distribution hypothesis, which means unbalanced and/or non-IID datasets.

2) Communication Costs: Federated learning is influenced by the rapidly growing depth of model and the amount of multiply-accumulate operations (MACs) [20]. This is due to the fact that the massive communication costs for uploading and downloading is necessary, while the average upload and download speed is asymmetric, e.g., 26.36 Mbps mean mobile download speed vs. 11.05 Mbps upload speed of UK in Q3-Q4 2017 [21].

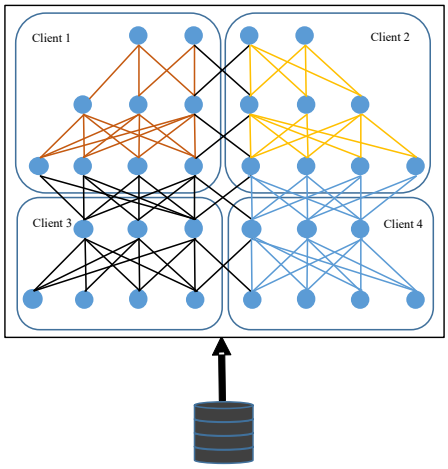


Fig. 1. The diagram of model parallelism. The complete model is distributed and stored in multiple clients, and the model is stitched after the training procedure is finished on the clients.

Obviously, high communication costs is one of the main reasons hindering distributed and federated training. Although the initial model compression related research is not intended to reduce the communication costs, it has been a source of inspiration for communication efficient distributed learning. Neural network pruning is an early method of model compression proposed in [22]. Parameter pruning and sharing in [22], [23], low-rank factorization [24], transferred/compact convolutional filters [25] and knowledge distillation in [26] are some main ideas reported in the literature. Reduction of communication costs by simultaneously minimazing the performance and minimizing the model complexity using an multi-objective evolutionary algorithm is reported in [27]. Recently, an layer-wise asynchronous model update approach has been proposed in [28] to reduce the number of parameters to be transmitted.

Gradient quantization has been proposed to accelerate data parallelism distributed learning [29]; gradient sparsification [30] and gradient quantization [31], [32] have been developed to reduce the model size; an efficient federated learning by sparse ternary compression (STC) has been proposed [33], which is robust to non-IID data and communication-efficient on both upstream and downstream communications. However, since the STC is a model compression method after local training is completed, the quantization process is not optimized during training.

To the best of our knowledge, most of the federated learning methods emphasize the application of full-precision networks or streamline models after the training procedure on client is finished, rather than simplifying the model during training. Therefore, deploying the federated learning environment in the widely used IoT devices in the process industry is somehow difficult. To address this issue, we focus on the model compression on clients during training to reduce the energy consumption at inference stage and the communication costs of federated learning. The main contributions of this paper are summarized as follows:

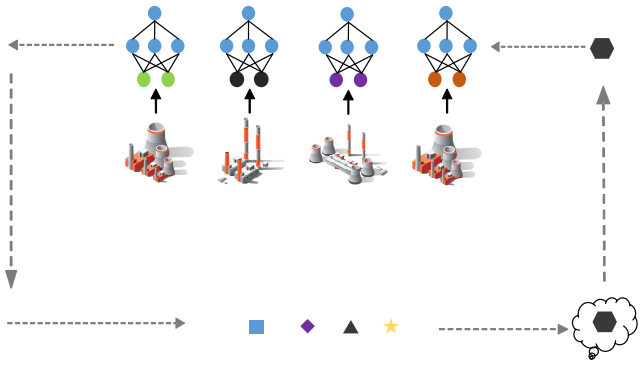


Fig. 2. An illustration to the application of federated learning in process industry. The branch factories train local models on private clients data through iterations, and send the trained local models to the server for aggregation to obtain optimized the global model.

- A ternary quantization approach is introduced into the training and inference stages of clients. The trained ternary models are well suited for inference in network edge devices, e.g., wireless sensors.
- A ternary federated learning protocol is presented to reduce the communication costs between clients and server, which compresses both upstream and downstream communications. Note that the quantification of the model weights can further enhance privacy protection since this makes the reverse engineering of the model more difficult.
- A theoretical analysis of the proposed algorithm is provided and the performance of the algorithm is empirically verified on a deep feedforward network (MLP) and a deep residual network (ResNet) using widely used datasets, namely MNIST [34] and CIFAR10 [35].

The remainder of this paper 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 quantify models of clients in federated learning systems, called federated trained ternary quantization (FTTQ), based on the quantitative algorithms mentioned earlier. On the basis of FTTQ, a ternary federated learning protocol that reduces both upstream and downstream communications is presented. In Section IV, a

theoretical analysis of the proposed algorithm is provided. Experimental settings and results are presented in Section V to compare the new protocol with standard algorithms. Finally, conclusions and future directions are given 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, followed by an numeric example.

A. Federated Learning Protocol

It is usually assumed that the data used by a distributed learning algorithm belongs to the same feature space, which may not be true in federated learning. As illustrated in Fig. 3, the basic protocol of federated learning proposed in [18] is round-based. Specifically, private storage, server and clients (usually mobile devices) are main participants in the whole protocol, and there are three main phases in each round: selection, configuration and reporting.

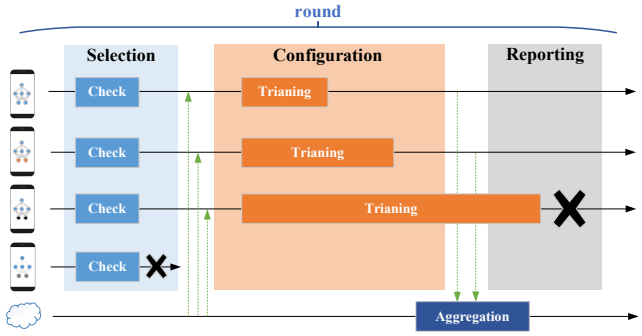


Fig. 3. Federated learning workflow with massive mobile devices. Firstly, the server selects suitable clients to deploy configuration (global model structure) for training. Then the clients complete the learning process in the specified budget (e.g., time) and return the local models to the server. Finally, the server aggregate all local models to obtain the trained global model.

In this work, we assume supervised learning is used for train the models. The global model with parameters θ, deployed in distributed client k, is trained based on local training dataset D_k, which consists of training sample pairs $(x_i,\ y_i)$, $(i = 1, 2, 3, ..., |D_k|)$. The loss of sample pair $(x_i,\ y_i)$ is denoted by $l(x_i, y_i; \theta)$, where l is the loss function. The total loss of a certain task on client k is J:

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

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

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

where λ is the proportion of the clients participating in the aggregation in the current round, which is determined by the above three phases.

Theoretically, the participation ratio λ in (2) is calculated by the number of participating clients and the total number of clients. Additionally, the local batch size B and local epochs E are also important hyperparameters. In the experiment, we will further study the effects of these parameters on the performance of the algorithm by manually setting the values of them.

It is easy to find that the communication costs are heavily dependent on the amount of information to be transferred between the server and clients, and the dominating factor in this procedure is the size of the parameters. One of the requirements for communication-efficient federated learning to be fulfilled is that both upstream and downstream communications need to be compressed [33]. Note that the performance of federated learning may dramatically drop due to disproportionation of data distribution.

B. Quantization

Quantization improves energy and space efficiency of deep networks by reducing the number of bits per weight [36]. 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. It has been shown that the ternary weight network (TWN for short) [32] is able to reduce the Euclidean distance between the quantization parameters and θ^t (consisting of -1, +1 and 0) and the full-precision parameters θ by a scaling factor α compared with binary networks [31], thus making the accuracy of the quantization network close to the full-precision network:

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

where F represents the cost function of this optimization problem; α^* and θ^{t*} are optimal solution to F, with $\theta \approx \alpha^*\theta^{t*}$. Li et al. [32] introduce an approximated optimal solution with a threshold-based function to quantify all layers of the deep neural network model:

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

where the θ_l and θ_l^t are the full-precision and quantized weights of l^{th} layer, and $\theta = \{\theta_1, \theta_2, ..., \theta_l, ...\}$, $\theta^t = \{\theta_1^t, \theta_2^t, ..., \theta_l^t, ...\}$, respectively, which provide a rule of thumb to calculate the optimal $\Delta^* = \{\Delta_1^*, \Delta_2^*, ..., \Delta_l^*, ...\}$.

However, the weights of TWN are limited to -1, 0, 1 and α^* is a constant. In order to further improve the performance of quantized deep networks while maintaining the compression ratio, Zhu et al. [37] have proposed a trained ternary quantization algorithm (TTQ for short). In TTQ, two quantization factors (positive factor w_p and negative factor w_n) are adopted to scale the ternary weights in each layer.

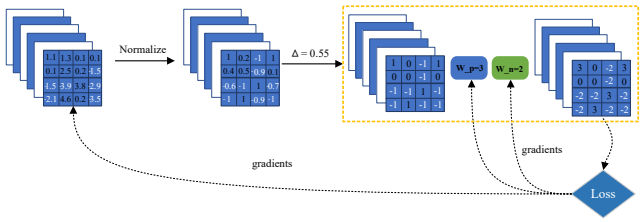


Fig. 4. An example of how the TTQ algorithm works. Firstly, normalized full-precision weights and biases are quantized to {-1, 0, +1} by the given threshold per layer. Secondly, positive and negative quantification factors are used to scale the quantized weights. Finally, the calculated gradients are back-propagated to each layer. The right part in the dotted rectangle represents the inference stage.

应该跟图4里面的符号对应

The workflow of TTQ is illustrated by Fig. 4, the normalized full-precision weights are quantized by Δ_l, w_l^p and w_l^n with full-precision activations. Instead of using the optimized threshold in TWN, TTQ adopts a heuristic method to calculate Δ_l:

$$\Delta_l = t \times \max(|\theta_l|), \tag{5}$$

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

III. Proposed Algorithm

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

A. Federated Trained Ternary Quantization

Since no direct data exchange is usually allowed between clients in the federated learning system, weight divergence [14] may be different among clients. For example in the l^{th} layer of the global model shared by client C_1 and client C_2, if $\max(|\theta_l^{C_1}|)$ = 5 and $\max(|\theta_l^{C_2}|)$ = 50, it is not necessarily true that the global model will be biased towards C_2 if we use the same factor for the two models. To address this issue, we start by scaling the weights to [-1, 1]:

$$\theta^s = g(\theta), \tag{6}$$

where g is a scaling function, $\mathbb{R}^n \rightarrow [-1, 1]$. However, magnitude imbalance [38] may be introduced when scaling the entire θ of a certain network, thus resulting in significant loss of precision, since most of the elements are pushed to zero. Therefore, we scale the weights layer by layer.

Then, by using the same strategy as TTQ, we calculate the quantization threshold Δ according to the scaled weights as follows:

$$\Delta = T_k \times \max(|\theta^s|), \tag{7}$$

where T_k is a hyperparameter with a default setting on the client k, and $\Delta = \{\Delta_1, \Delta_2, ..., \Delta_l, ...\}$.

However, according to (6) and (7), we can easily find that the thresholds in all layers are mostly the same since the maximum absolute value of the scaled θ^s is 1 in most layers. Thus, the model capability may be effected by the homogeneity of the threshold. To avoid this issue, we propose an alternative threshold calculation criterion:

$$\Delta = \frac{T_k}{m} \sum_i^m (|\theta_i^s|), \tag{8}$$

where m is the number of neurons and Δ is layer-wise calculated. Obviously, the threshold obtained by (8) is influenced by the layer sparsity and can be seen as an extension of (7) as:

$$\begin{aligned} \Delta = T_k \times \frac{1}{m} \sum_i^m (|\theta_i^s|) &\leq T_k \times \frac{1}{m} (m \times \max|\theta^s|) \\ &\leq T_k \times \frac{1}{m} (m \times 1) \leq T_k. \end{aligned} \tag{9}$$

Notably, the threshold will turn into the optimal solution proposed in [32] if we set the value of T_k to 0.7. The calculation method of Δ is generally adjusted according to the performance.

Subsequently, several operations are taken to achieve layer-wise weight quantization to overcome the computation burden and reduce the communication costs:

$$mask = \varepsilon(|\theta^s| - \Delta), \tag{10}$$

$$I^t = \text{sign}(mask \odot \theta^s), \tag{11}$$

$$\theta^t = w^q \times I^t, \tag{12}$$

where ε is the step function and $\odot$ is the Hadamard product, $w^q = [w_1^q, w_2^q, ..., w_l^q, ...]$ is an independent quantization vector which is trained together with other parameters layer by layer, and I^t is the quantized ternary weights. Consequently, the mask matrix can be rewritten as a union of a positive index I^p and a negative index I^n of the local model:

$$I_p = \{i \mid \theta_i > \Delta\}, \tag{13}$$

$$I_n = \{i \mid \theta_i < -\Delta\}. \tag{14}$$

Different from the standard TTQ, we adopt a quantization factor which is updated with its gradients together with other parameters instead of the previous two quantization factors in each layer, mainly due to the following reasons.

1) Stability: Large weight divergence will be encountered after synchronization if participating clients are initialized with different parameters, which leads to performance degeneration [14]. Hence, the weight divergence should be minimized at each layer in the federated learning environment.

2) Energy Consumption: We present an proposition about the convergence trends of w_p and w_n and its proof in Section IV, followed by the experimental results on the two factors with different initials when training MLP and ResNet* in Appendix A. It is worth noting that the trend of the positive and negative quantization factors of TTQ algorithm is almost the same in all layers. Hence, the energy consumption in calculating the gradients of two quantization factors during the back propagation can be cut in half if only one quantization factor is retained, which is important for some resource-constrained clients.

After quantifying the whole network, the loss function can be calculated and the errors be backpropagated in the same way as for continuous weights except that the weights are $\pm w^q$ or zeros. The gradients of w^q and latent full-precision model are calculated according to the rules in [37]. The new update rule is summarized in Algorithm 1. 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 measures are taken, since the weights of the global model cannot be decomposed into the coefficient and ternary matrix after aggregation. To address this issue, a ternary federated learning protocol is presented in the next section.

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: Quantified model θ^t
init: All clients parameters are initialized with θ.
for $(x_i, y_i) \in D$ **do**
  $\theta^s \leftarrow g(\theta)$
  $mask \leftarrow \varepsilon(|\theta^s| - \Delta)$
  $I^t = \text{sign}(mask \odot \theta^s)$
  $\theta^t \leftarrow w^q \times I^t$
  $J \leftarrow l_i(x_i, y_i; \theta^t)$
  $\frac{\partial J}{\partial w^q} \leftarrow \sum_{i \in I_p} \frac{\partial J}{\partial \theta_i^t}$
  $w^q \leftarrow w^q + \eta \frac{\partial J}{\partial w^q}$
  $\theta^t \leftarrow \theta^t + \eta \frac{\partial J}{\partial \theta^t}$
  update θ
end
Return θ^t (including w^q, I^t)

B. Ternary Federated Averaging

The two-step scheme of the proposed ternary federated averaging protocol with private data is elaborated in Fig. 5. In general, the participating clients quantize the normalized local models and upload the thresholds, quantization factors and ternary models to the server. Then the server aggregates all local models to obtain the global model. Finally, the server quantifies the normalized global model again using fixed thresholds and pushes the quantized global model to all clients. The basic flow is described as follows.

1) Upstream: Let $\mathbb{K} = \{1, 2, ..., |\lambda N|\}$ be the set of indices which represent the randomly selected clients that participate in the aggregation, where λ is the participation ratio and N is the total number of the clients in the federated learning system. The local scaled 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 instead of the updates $\nabla\theta_k$ after local iterations, although the two are equivalent [11]. And at inference stage, only the quantized model is needed for prediction.

2) Downstream: After r communication rounds, the server will rebuild all models received from the participated clients, and the global model can be calculated by $\theta_{r+1} \leftarrow \sum_{k=1}^{\lambda N} \frac{|D_k|}{\sum_{k=1}^{\lambda N} |D_k|} \theta_k^t$. Then the server will quantify the global model again with a constant threshold Δ using a default setting of 0.05 and push the quantized model to the clients.

Algorithm 2: Ternary Federated Averaging

Input: Initial global model parameters θ
Init: Broadcast θ to clients k, $k = \{1, 2, 3, ..., N\}$, assign each client a unique dataset D_k.
for *round* r = 1,..., T **do**
  for $c \in \mathbb{K} = \{1, 2, ..., |\lambda N|\}$ **in parallel do**
    Client k **does:**
      download quantified θ_{r-1}^t
      initialize w^q
      $\theta_{k,r}^t \leftarrow$ **FTTQ**(θ_{r-1}^t, w^q)
      upload $\theta_{k,r}^t$ to server
    end
  end
  Server does:
    $\theta_r \leftarrow \sum_{k=1}^{\lambda N} \frac{|D_k|}{\sum_{k=1}^{\lambda N} |D_k|} \theta_{c,r}^t$
    $mask \leftarrow \varepsilon(|\theta_r| - \Delta)$
    $\theta_r^t \leftarrow \text{sign}(mask \odot \theta_r)$
    broadcast θ_r^t to all clients
  end
end

Unlike standard federated learning algorithms, our method compresses communications during the upload and download phases, which brings major advantages when deploying DNNs at the inference stage for resource-constrained devices. Specifically, the clients move the local networks from 32-bit to 2-bit and push the 2-bit networks and quantification parameters 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

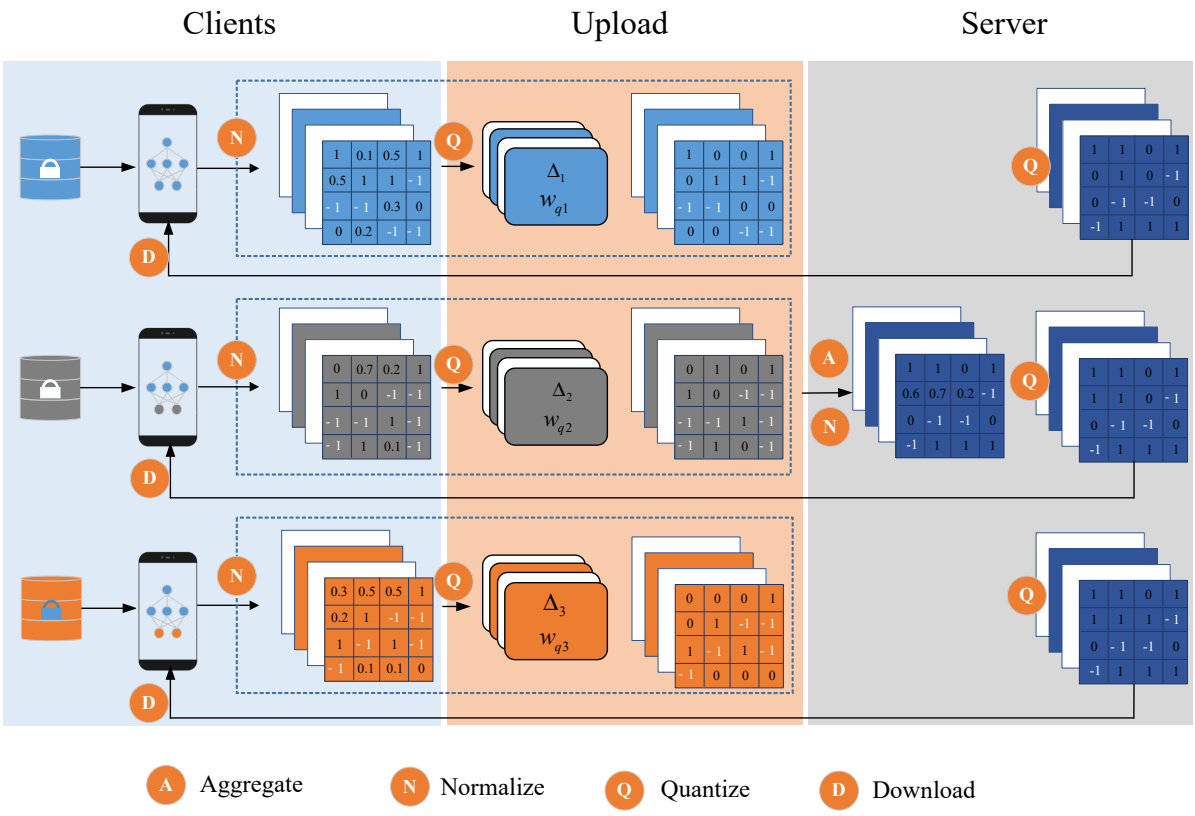


Fig. 5. The diagram of proposed T-FedAvg. The blue part runs on the clients with normalized full-precision weights, which is similar to the standard federated learning framework; then the quantification factors, thresholds and ternary local models are pushed to the server, as shown in the orange part; after that, the global model is obtained by the server aggregation and normalization; finally, the global model is quantized and pushed back to all clients.

the costs to 65 MB per round (upload and download), which is about 1/16 of the standard method. Note that quantifying the global model pushed to the clients makes reverse engineering more difficult. The overall workflow of the proposed ternary federated protocol is summarized in Algorithm 2.

IV. Theoretical Analysis

In this section, we first formally demonstrate the properties of two quantization factors w_p, w_n in TTQ, followed by a proof of unbiasedness of FTTQ and T-FedAvg.

By default in this paper, the subscripts represent the indices of the elements in a network instead of the indices of the clients in the federated learning system.

A. The Convergence of Quantization Factors in TTQ

The experimental results of the convergence profiles of w_p and w_n of two widely used neural networks are presented in Appendix A, showing that the two factors have converged to the same value. To theoretically prove the convergence, we at first introduce the following assumption.

Assumption 4.1: The elements in scaled full-precision θ are uniformly distributed between 0 and 1,

$$\forall \theta_i \in \theta, \theta_i \backsim U(-1,1). \tag{15}$$

Then we have the following proposition.

Proposition 4.1: Given an one-layer online gradient system, each element of its parameters is initialized with a symmetric distribution centered at 0, e.g., $\theta_i \backsim U(-1,1)$, which is quantized by TTQ with two iterative factors w_p, w_n and a fixed threshold Δ, then we have:

$$\lim_{e\to+\infty} w_p = \lim_{e\to+\infty} w_n, \tag{16}$$

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

Proof 4.1: 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:

$$w_p^*, w_n^* = \arg\min_{w_p, w_n} \|\theta - w_p I_p + w_n I_n\|_2^2, \tag{17}$$

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

$$\theta - w_p I_p + w_n I_n = \begin{cases} \theta_i - w_p, & i \in I_p \\ \theta_k, & k \in I_z \\ \theta_j + w_n, & j \in I_n. \end{cases} \tag{18}$$

Then the original problem can be transformed to

$$\begin{aligned} &\|\theta - w_p I_p + w_n I_n\|_2^2 \\ &= \sum_{i\in I_p}(\theta_i - w_p)^2 + \sum_{j\in I_n}(\theta_j + w_n)^2 + \sum_{k\in I_z}\theta_k^2 \\ &= |I_p| w_p^2 + |I_n| w_n^2 - 2w_p \sum_{i\in I_p}\theta_i + 2w_n \sum_{j\in I_n}\theta_j + C, \end{aligned} \tag{19}$$

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 (19) can be obtained when

$$\begin{aligned} w_p^* &= \frac{1}{|I_p|}\sum_{i\in I_p}\theta_i, \\ w_n^* &= -\frac{1}{|I_n|}\sum_{j\in I_n}\theta_j. \end{aligned} \tag{20}$$

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

B. The Unbiasedness of FTTQ

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

With Assumption 4.1, we prove Proposition 4.2:

Proposition 4.2: Let θ be the local scaled network parameters defined in Assumption 4.1 of one client in a given federated learning system. If θ is quantified by FTTQ algorithm, then we have

$$E\left[FTTQ(\theta)\right] = E\left(\theta\right). \tag{21}$$

Proof 4.2: According to (20), w_q^* is calculated by the elements in $I_p = \{k|\theta_k \geq \Delta\}$, where Δ is a fixed number once the parameters are generated under Assumption 4.1, hence the elements indexed by I_p obey a new uniform distribution between Δ and 1, then we have

$$\forall k \in I_p, \theta_k \backsim U(\Delta, 1), \tag{22}$$

therefore, the probability density function f of θ_k $(k \in I_p)$ can be regarded as $f(x) = \frac{1}{1-\Delta}$.

According to Proposition 4.1 and (20), we have:

$$\begin{aligned} E\left(w^{q*}\right) &= E\left(\frac{1}{|I_p|}\sum_{k\in I_p}\theta_k\right) = \frac{1}{|I_p|}E\left(\sum_{k\in I_p}\theta_k\right) \\ &= \frac{1}{|I_p|}|I_p|\int_{\Delta}^{1}\theta_i f(\theta_i) = \int_{\Delta}^{1}\theta_i f(\theta_i) \\ &= \frac{1+\Delta}{2}, \end{aligned} \tag{23}$$

where θ_i is an arbitrary element in I_p and $|I_p|$ represents the number of elements in I_p.

We know that

$$\begin{aligned} E\left[FTTQ(\theta)\right] &= E\left[w^{q*} \times sign(mask(\theta) \times \theta)\right] \\ &= E(w^{q*})E\{sign(\theta)E\left[mask(\theta)\right]\}, \end{aligned} \tag{24}$$

and since

$$\begin{aligned} E\left[mask(\theta)\right] &= P\left[mask(\theta) = 1)\right] \times 1 + P\left[mask(\theta) = 0\right] \\ &\times 0 + P\left[mask(\theta) = 0\right] \times (-1) \\ &= \frac{1-\Delta}{2} \times 1 + \Delta \times 0 + \frac{1-\Delta}{2} \times (-1) \\ &= 0, \end{aligned} \tag{25}$$

hence

$$\begin{aligned} E\left[FTTQ(\theta)\right] &= E(w^{q*})E\{sign(\theta)E\left[mask(\theta)\right]\} \\ &= \frac{1+\Delta}{2} \times 0 = 0, \end{aligned} \tag{26}$$

and under Assumption 4.1, we have

$$E\left(\theta\right) = \int_{-1}^{1}\theta d\theta = 0, \tag{27}$$

then it is immediate that

$$E\left[FTTQ(\theta)\right] = E\left(\theta\right). \tag{28}$$

Hence, the FTTQ quantizer output can be considered as an unbiased estimator of the input [39]. We can guarantee the unbiasedness of FTTQ in federated learning systems when the weights are uniformly distributed. Furthermore, since the distribution of network weights in most of layers may be non-uniform due to the stochastic errors from data, the self-learning factor w^q and Δ may reduce the quantization errors, which can be regarded as a non-uniform sampling method.

C. The Properties of T-FedAvg

Here, we adopt the following assumption to demonstrate the properties of T-FedAvg, which is widely used rules in the literature [14].

Assumption 4.2: When a federated learning system with K clients and one server is established, all clients will be initialized with the same global model.

Under the above assumption, Zhao et al. [14] proved the following conclusions:

Lemma 4.1: The weight divergence which leads to the performance degeneration after r rounds of synchronization between the clients and the server mainly comes from two parts, including the weight divergence after $r-1$ rounds of aggregation, i.e., $||\theta_{r-1}^f - \theta_{r-1}^c||$ (superscript c denotes the centralized setting), and the weight divergence resulting from the Earth mover's distance (EMD) between the data distribution on each client and the actual distribution of the whole data population.

Lemma 4.2: When all the clients are initialized with the same global model, the EMD between the data distribution on each client and the distribution of the whole data population becomes the main cause of the performance degeneration.

Since our method is unbiased and can reduce the Euclidean distance between the quantized network and the full-precision network, we can conclude that T-FedAvg can also perform well if the original FedAvg converges to the optimal solution.

V. Experimental Results

This section evaluates the performance of the proposed method on widely used benchmark datasets. 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 10 independent physical clients connected by a Local Area Network (LAN). Then, we test the obtained model in the simulation environment with the number of clients varying from 10 to 100.

The physical system consists of four CPU laptops that are connected wirelessly through LAN to mimic low-power mobile devices, one of which acts as the server aggregation model and the remaining laptops 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 10 clients for experiments according to the number of classes in the datasets. A detailed description of the configuration is given below.

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

- Baseline: the centralized learning algorithm, such as stochastic gradient descent (SGD) method, which means that all data is stored in a single computing center and the model is trained directly using the entire data.
- FedAvg: the canonical federated learning approach presented in [11].
- TTQ: the canonical trained ternary quantization method, in which the configuration is the same as the baseline, i.e., the data is stored in a centralized manner and a model is trained using all the data.
- T-FedAvg: our proposed quantized federated learning approach.

2) **Datasets**. We select two representative benchmark datasets that are widely used for classification, and no data augmentation method is used in all experiments.

- MNIST [34]: it contains 60000 training and 10000 testing gray-scale handwritten image samples with 10 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.
- CIFAR10 [35]: it contains 60000 colored images of 10 types of objects from frogs to planes, 50000 for training and 10000 for testing. It is a widely used benchmark data set that is difficult to extract features.

2) **Models**. To evaluate the performance of above algorithms, two popular deep learning models are selected: MLP and ResNet*, which represent tiny and large models, respectively. The detailed setting are as follows:

- MLP: it is mainly used for training small data sets, e.g., MNIST. The model contains two hidden layers with the number of neurons of 30 and 20, respectively. For centralized and distributed training, the learning rate η is set to the same and the ReLU function is selected as the activation function.
- ResNet18*: it is a simplified version of the widely used ResNet [40], 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.

3) **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:

- IID data: each client holds an IID subset of data containing 10 classes, thus having a IID-subset of the data.
- Non-IID data: the union of the samples in all clients is the entire dataset, but the number of classes contained in each client is not equal to the total number of categories in the entire dataset (10 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.
- Unbalancedness in data size: typically, the size of the datasets 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 dataset into several distinct parts.

3) **Basic configuration**. The basic configuration of federated learning system in our experiments is set as follows:

- Total number of clients: N = 100.
- The participation ratio per round: λ = 0.1.
- Classes per client: N_c = 10.
- Local batch size: B = 64.
- Local epochs: E = 5.

All experimental settings are summarized in Table I.

TABLE I
MODELS AND HYPERPARAMETERS.

Models	MLP	ResNet*
Dataset	MNIST	CIFAR10
Optimizer	SGD	Adam
Learning rate	0.0001	0.008
Baseline	92.75%	86.30%
Parameter amount	24330	607050

The learning rate of the centralized and federated learning algorithms is the same and remains constant during the training. Note that a small learning rate is set for training MLP to slow down the convergence speed for easy observation.

B. Performance on IID Data

In this part, we conduct experiments on IID MNIST and CIFAR10 using the benchmark algorithms with MLP and ResNet* mentioned above, where the baseline and TTQ are representatives of centralized approaches.

Specifically, the data used by the centralized methods is stored centrally in one computing center, while the data used by FedAvg and T-FedAvg is stored separately. To explore the best performance of FedAvg and T-FedAvg, the federated learning environment is set with 10 fully participating clients and each client holds an IID subset of data containing 10 classes.

The results and model weight width are summarized in Table II. We can see that the test accuracies achieved by the baseline algorithm and TTQ are 92.75%, 92.87%, respectively, when trained on MNIST with MLP, and 86.30%, 85.73%, respectively, on CIFAR10 with ResNet*. TTQ has a slight deterioration in performance on CIFAR10 when the model complexity is reduced.

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

Methods	MNIST		CIFAR10	
	Accuracy	Width	Accuracy	Width
Baseline	92.75%	32 bit	86.30%	32 bit
FedAvg	92.37%	32 bit	85.72%	32 bit
TTQ	92.87%	2 bit	85.73%	2 bit
T-FedAvg	92.75%	2 bit	86.60%	2 bit

When the data distribution is IID, FedAvg achieves 92.37% and 85.72% accuracies, respectively, on MNIST and CIFAR10. However, the test accuracy of T-FedAvg, which is about 1/16 of the full-precision model size, achieves 92.75% on MNIST and the highest on CIFAR10, 86.60%. It is worth noting that, as the depth of the network deepens, the quantization error is declining and may even exceed the accuracy of the original model, which can be evidenced by the performance of T-FedAvg on ResNet.

The convergence speeds of the four methods in different local iterations are illustrated in Fig. 6, where the centralized methods, the baseline and TTQ obtain convergence curves corresponding to the federated learning environment by the interval of $\lambda * N * E$. Overall, the convergence speed of our method is the fastest when trained on MNIST and is slightly slower than FedAvg on CIFAR10 in the initial phase, which is up to the performance of TTQ.

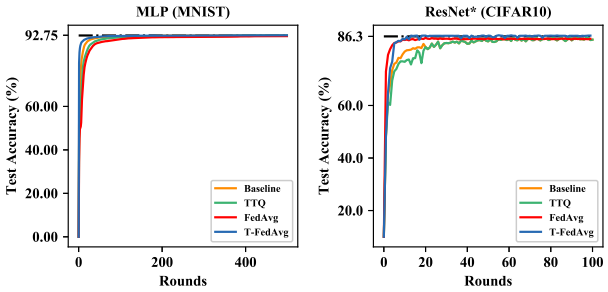


Fig. 6. Convergence speed of the four compared algorithms over communication rounds or epochs with the same models.

The test accuracies achieved by FedAvg and T-FedAvg for various batch sizes are shown in Fig. 7. We notice that our method outperforms FedAvg for a small batch size. Since small batches mean more iterations and can thus reduce quantization errors, it is beneficial for clients with limited computing resources. However, the performance of T-FedAvg is not robust enough compared to FedAvg in big local batch size. This may be attributed to the fact that insufficient model training has resulted in an accumulation in quantization errors when the local batch size is large.

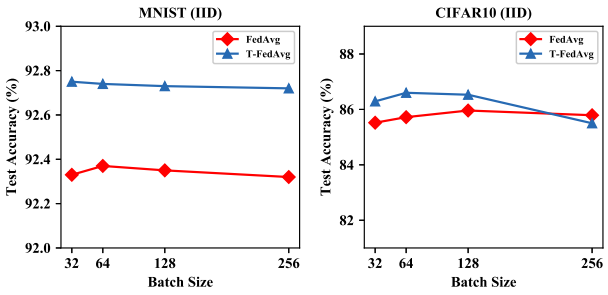


Fig. 7. Maximum accuracies achieved by FedAvg and T-FedAvg when training MLP on MNIST (left) and ResNet18* on CIFAR10 (right), respectively, for 100 rounds with different batch sizes. Ten clients are involved for all experiments in full participation.

C. Performance with Different N_c

The boxplots of data distributions with different N_c are depicted in Fig. 9, where the y-axis represents the sample label (0-9). As shown in the figure, the original distribution of training and test data are IID. Specifically, the data distribution is IID in the case of N_c = 10, which means that each client has a IID subset of the entire dataset (refer to the plot right). However, when the N_c = 2, the samples on each client are divided according to the label, which is non-IID, and has no overlap with other clients. Similarly, the samples in all clients are non-IID when N_c = 5, but there are some overlaps in data between clients. Clearly, the data distribution can be regarded as non-IID when N_c is smaller than the number of total classes categories of the training data. In this case, the local stochastic gradient cannot be considered as an unbiased estimate of the global gradient with non-IID settings.

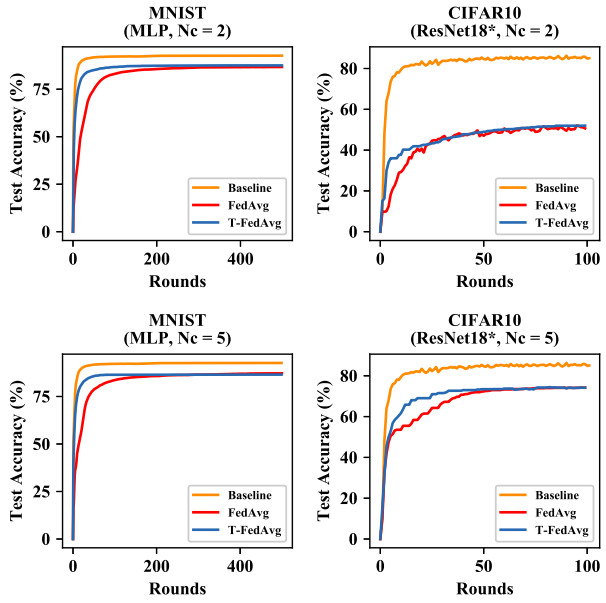


Fig. 8. Test accuracies achieved by MLP and ResNet* trained on non-IID MNIST and CIFAR10 after fixed rounds of FedAvg and T-FedAvg. The participation ratio is fixed to 1 and the training data is split among the clients with different N_c.

Although FedAvg and T-FedAvg have achieved satisfactory test accuracies on IID data, a significant degradation of the test performance of FedAvg and T-FedAvg is observed on non-IID data, which is illustrated in Fig. 8 and Table. III. Note that 10 clients are selected with full participation and no pre-training model is used during the training, which means that the participating ratio λ is 1 for the purpose of investigating the impact of N_c for the non-IID setting.

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

Methods	MNIST		CIFAR10	
	$N_c = 2$	$N_c = 5$	$N_c = 2$	$N_c = 5$
FedAvg	86.69%	87.17%	52.10%	74.21%
T-FedAvg	87.1%	87.22%	52.35%	74.43%

However, as mentioned in the previous work [11], federated learning suffers from extremely non-IID data distribution. When $N_c = 2$ and 5, each client is randomly assigned 2 and 5 classes, respectively, both methods work well on MNIST, although there is an acceptable performance degradation. Nevertheless, a significant reduction in the test accuracy on CIFAR10 is observed for both methods when $N_c = 2$, and increasing N_c from 2 to 5 can effectively alleviate the degeneration.

As we all know, the intricate features of CIFAR10 makes it more difficult to train the model than MNIST. Therefore, the performance gap on MNIST when $N_c = 2$ and 5 is smaller than that on CIFAR10. And during the training process, T-FedAvg outperforms the standard FedAvg in terms of convergence speed, and is similar to that of the baseline method at the earlier stage.

Theoretically, since T-FedAvg could reduce the upstream and downstream communication costs, we can increase the number of clients or communication rounds within the same constraint of budgets to alleviate the performance degeneration. Recently, a method which shares partial selected IID data to alleviate the performance degeneration has also been proposed [14]. Although the method has certain limitations (e.g., the way of generating the shared dataset and the over-fitting problems introduced by the shared dataset), it is still an promising solution.

D. Influence of the Participation Ratio λ

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 Fig. 8) can reduce the effect of model selection. Fig. 10 shows 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).

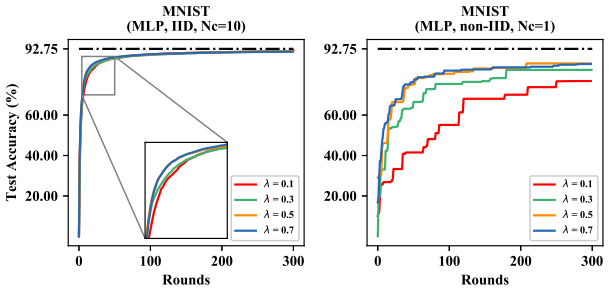


Fig. 10. Test accuracies achieved by T-FedAvg when training MLP on MNIST with IID distribution (left) and non-IID distribution (right) in fixed rounds at different participation ratios (0.1, 0.3, 0.5, 0.7).

As we can see, T-FedAvg is relatvely robust to the changes of the participation ratio λ over IID and non-IID data. Although reducing participation ratios λ has negative effects on the convergence speed and convergence values achieved in the fixed rounds, the negative effects are more pronounced on non-IID data (refer to the right panel of Fig. 10). A similar phenomenon has also been observed in [33]. We surmise that the performance degradation on non-IID data is heavily dependent on whether the features on the clients selected by the server for model aggregation are representative in the federated learning environment. It is common to increase the participation ratio to alleviate the negative impacts.

E. Influence of Unbalancedness in Dada 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 [33]. If we use $S_N = \{|D_1|, |D_2|, ..., |D_N|\}$ to represent

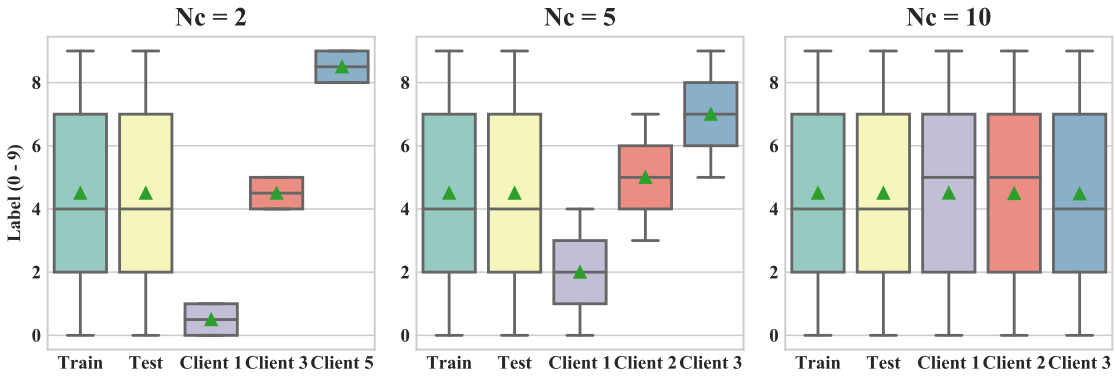


Fig. 9. Data distributions with different N_c. When N_c =2, there is no overlap in data between clients and each client contains two categories (left). When N_c =5, the samples on 10 clients are sampled by labels but there are some overlap between clients (middle). When N_c =10, the samples on 10 clients are generated by randomly sampling (right). Note that Only 3 clients are shown in the figures when the N_c is 2, 5 or 10.

the set of number of samples on N clients, we can define the degree of unbalanceness by the ratio β:

$$\beta = \frac{median\{S_N\}}{max\{S_N\}}, \tag{29}$$

where the median of S_N is sometimes helpful to accommodate long tailed distributions and possible outliers [41].

When β = 0.1, most of the samples are stored on a few clients, and when β = 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 participating. And the test accuracies achieved by FedAvg and T-FedAvg for various β are illustrated in Fig. 11.

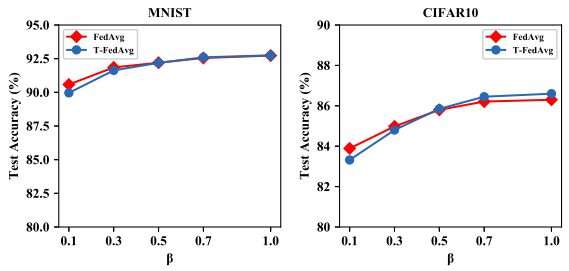


Fig. 11. Test accuracies achieved by MLP on MNIST and ResNet* on CIFAR10 after 400 rounds of iterations with FedAvg and T-FedAvg, where the local batch size and participation ratio are set to 32 and 0.3, respectively.

We can see that the unbalancedness in data size does not have a significant impact on the performance of federated learning. This is due to the fact that even when the data is mostly concentrated on some clients, both algorithms can achieve satisfactory performance.

F. Comparison of Communication Costs

In this subsection, we compared the communication costs of FedAvg and T-FedAvg for a fixed round number. The learning environment is configured the same as in Section IV. Since both algorithms have achieved convergence within 100 rounds on all datasets, we fix the round number to 100. The results are shown in Table IV.

TABLE IV
THE TOTAL MEMORY REQUIRED TO ACHIEVE A CERTAIN TARGETED TEST ACCURACY ON DIFFERENT TASKS IN AN IID SETTING WITHIN 100 ROUNDS. NOTE THAT 10 OUT OF 100 CLIENTS (λ = 0.1) PARTICIPATING THE AGGREGATION AFTER 5 LOCAL TRAINING EPOCHS IN ONE ROUND.

Methods	MLP		ResNet*	
	Upload	Download	Upload	Download
FedAvg	742.49 MB	742.49 MB	18525.70 MB	18525.70 MB
T-FedAvg	46.41 MB	46.41 MB	1157.86 MB	1157.86 MB

We can see that the communication costs of T-FedAvg are reduced by nearly 94% in the upload and download phases compared to the standard FedAvg. To the best of our knowledge, no such a significant communications compression level in the download phase has been achieved in federated learning.

VI. Conclusions 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 extreme. To address these issues, we have proposed federated trained ternary quantization (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 to nearly one sixteenth of the standard method. The optimal solutions of the quantization factors, detailed proofs of the unbiasedness and convergence of the proposed methods are also given. Our experimental results on widely used benchmark datasets demonstrate the effectiveness of the proposed algorithms. Moreover, since we have reduced the downstream and upstream communication costs between the clients and the 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 global model to reduce the communication costs. However, the great reduction in communication costs is at the expense of the performance of federated learning, in particular when the data on the clients are extremely non-IID. Our future work will aim at finding a more efficient approach to improving the performance of federated learning on non-IID data.

Appendix A
Trade-off between Model Capacity and Communication Costs

In this appendix, we analyze the weight distribution and the convergence of the quantification factors in the proposed TTQ algorithm. Both MLP and ResNet* are selected to enhance the reliability of the analysis. The initial values of the quantization factors are changed in the case of fixing other hyperparameters (e.g., the batch size), and the influences of the quantization factors are observed from two aspects: the convergence trend in each layer and the effect of the absolute difference in the quantization factor values.

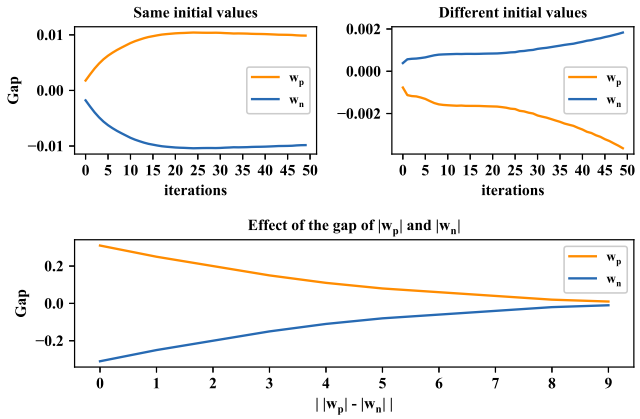


Fig. 12. Convergence analysis of quantization factors in ternary MLP. The convergence trend under different initial values (top) and the effect of the gap of positive and negative values (bottom).

Firstly, we conduct experiments on MLP since there exists only one quantized hidden layer. As shown on the top of Fig. 12, if we subtract the initial value from the values obtained by each iteration of the quantization factor, we can see that the gap between the iterative values and the initial values of the positive and negative factors changes in the same trend. Regardless of whether the initial values are the same, the offsets are consistent with respect to the respective initial values.

As we know, if the convergence values are the same as the initial values, TTQ degenerates into TWN and the learning capability of ternary models may decline. Especially, $\max(|\theta|)$ varies greatly due to the distinct data distribution of each client in the process of federated learning. To address this issue, we reduce the two quantization factors to one, and use (6), (8) to constrain the threshold between clients.

To illustrate the effect of the gap of positive and negative values, we fix the initial value of one of the factors and increase the initial value of the other factor, as shown in the bottom of Fig. 12. We can see that as the interval increases, the change in the values of quantification factors is smaller, and finally it is almost close to 0 (convergence to the initial values). So we can conclude that the gradients of I_l^p and I_l^n will be tiny in the end, according to (13) and (14).

Similar phenomena can also be observed from the experiments conducted on ResNet*. The results obtained by the selected appropriate initial values are shown in Fig. 13. When the initial values of the two quantization factors are the same, the convergence profile of w_l^p and w_l^n in the l^{th} layer is nearly symmetrical and the difference between the absolute values of the two is almost zero at each epoch (refer to Fig. 13(a)). In case the initial values of the w_l^p and w_l^n are different, it can be observed that w_l^p and w_l^n have the same trend in the l^{th} layer from Fig. 13(b), while the convergence trend of the two parameters is more fluctuating.

References

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

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.

[3] P. Voigt and A. Von dem Bussche, "The eu general data protection regulation (gdpr)," *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 2017.

[4] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le *et al.*, "Large scale distributed deep networks," in *Advances in neural information processing systems*, 2012, pp. 1223–1231.

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

[6] A. Smola and S. Narayanamurthy, "An architecture for parallel topic models," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 703–710, 2010.

[7] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous systems, 2015," *Software available from tensorflow. org*, vol. 1, no. 2, 2015.

[8] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server," in *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14)*, 2014, pp. 583–598.

[9] S. Zhang, C. Zhang, Z. You, R. Zheng, and B. Xu, "Asynchronous stochastic gradient descent for dnn training," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 6660–6663.

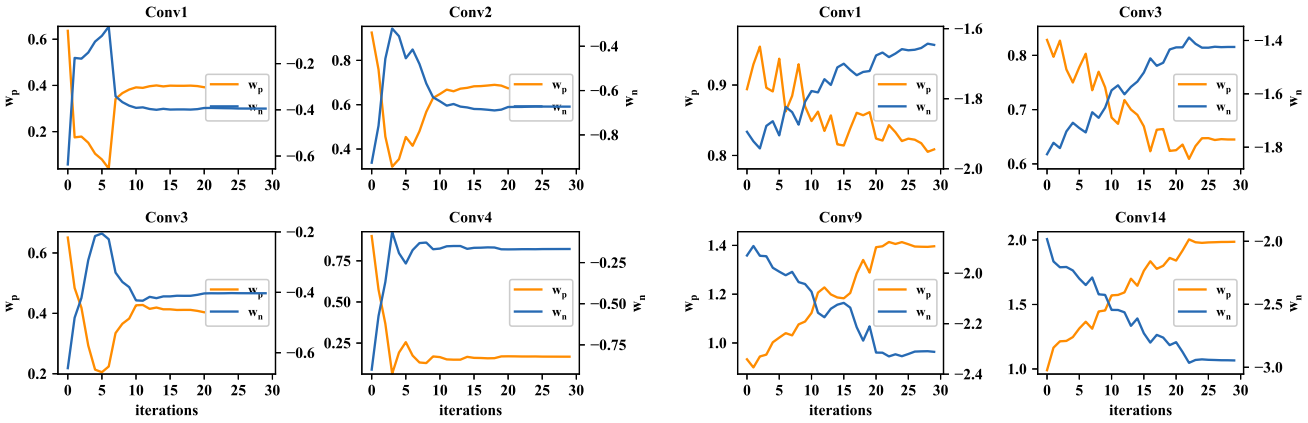
[10] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G. R. Ganger, P. B. Gibbons, and O. Mutlu, "Gaia: Geo-distributed machine learning approaching {LAN} speeds," in *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, 2017, pp. 629–647.

[11] H. B. McMahan, E. Moore, D. Ramage, S. Hampson *et al.*, "Communication-efficient learning of deep networks from decentralized data," *arXiv preprint arXiv:1602.05629*, 2016.

[12] J. Konečnỳ, B. McMahan, and D. Ramage, "Federated optimization: Distributed optimization beyond the datacenter," *arXiv preprint arXiv:1511.03575*, 2015.

[13] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016.

[14] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.



(a) Same initial values

(b) Different initial values

Fig. 13. The convergence trend in specific layer and convergence values among layers with same (left) and different (right) initial values of the quantification factors.

[15] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *learning*, vol. 8, p. 9, 2018.

[16] Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, and B. He, "A survey on federated learning systems: Vision, hype and reality for data privacy and protection," *arXiv:1907.09693 preprint*, 2019.

[17] •, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 2, p. Article No. 12, 2019.

[18] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecny, S. Mazzocchi, H. B. McMahan *et al.*, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.

[19] W. Lim, N. C. Luong, D. T. Hoang, Y.-C. L. Yutao Jiao, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *arXiv:1909.11875*, 2019.

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

[21] speedtest.net, "United kingdom mobile speedtest data," *https://www.speedtest.net/reports/united-kingdom/*, 2016.

[22] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in neural information processing systems*, 1990, pp. 598–605.

[23] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in neural information processing systems*, 2015, pp. 1135–1143.

[24] R. Rigamonti, A. Sironi, V. Lepetit, and P. Fua, "Learning separable filters," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 2754–2761.

[25] T. Cohen and M. Welling, "Group equivariant convolutional networks," in *International conference on machine learning*, 2016, pp. 2990–2999.

[26] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[27] H. Zhu and Y. Jin, "Multi-objective evolutionary federated learning," *IEEE transactions on neural networks and learning systems*, 2019.

[28] Y. Chen, X. Sun, and Y. Jin, "Communication-efficient federated deep learning with layer-wise asynchronous model update and temporally weighted aggregation," *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

[29] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li, "Terngrad: Ternary gradients to reduce communication in distributed deep learning," in *Advances in neural information processing systems*, 2017, pp. 1509–1519.

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

[31] M. Courbariaux, Y. Bengio, and J. P. David, "Binaryconnect: training deep neural networks with binary weights during propagations," in *International Conference on Neural Information Processing Systems*, 2015.

[32] F. Li, B. Zhang, and B. Liu, "Ternary weight networks," *arXiv preprint arXiv:1605.04711*, 2016.

[33] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *arXiv preprint arXiv:1903.02891*, 2019.

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

[35] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[36] 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," *arXiv preprint arXiv:1606.06160*, 2016.

[37] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," *arXiv preprint arXiv:1612.01064*, 2016.

[38] A. Polino, R. Pascanu, and D. Alistarh, "Model compression via distillation and quantization," *arXiv preprint arXiv:1802.05668*, 2018.

[39] R. M. Gray and D. L. Neuhoff, "Quantization," *IEEE transactions on information theory*, vol. 44, no. 6, pp. 2325–2383, 1998.

[40] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[41] A. W. Bowman and A. Azzalini, *Applied smoothing techniques for data analysis: the kernel approach with S-Plus illustrations*. OUP Oxford, 1997, vol. 18.