Communication-Efficient Federated Learning with Binary Neural Networks

Yuzhi Yang, Student Member, IEEE, Zhaoyang Zhang, Senior Member, IEEE, and Qianqian Yang, Member, IEEE

Abstract—Federated learning (FL) is a privacy-preserving machine learning setting that enables many devices to jointly train a shared global model without the need to reveal their data to a central server. However, FL involves a frequent exchange of the parameters between all the clients and the server that coordinates the training. This introduces extensive communication overhead, which can be a major bottleneck in FL with limited communication links. In this paper, we consider training the binary neural networks (BNN) in the FL setting instead of the typical real-valued neural networks to fulfill the stringent delay and efficiency requirement in wireless edge networks. We introduce a novel FL framework of training BNN, where the clients only upload the binary parameters to the server. We also propose a novel parameter updating scheme based on the Maximum Likelihood (ML) estimation that preserves the performance of the BNN even without the availability of aggregated real-valued auxiliary parameters that are usually needed during the training of the BNN. Moreover, for the first time in the literature, we theoretically derive the conditions under which the training of BNN is converging. Numerical results show that the proposed FL framework significantly reduces the communication cost compared to the conventional neural networks with typical real-valued parameters, and the performance loss incurred by the binarization can be further compensated by a hybrid method.

Index Terms—Federated learning, binary neural networks (BNN), Maximum Likelihood (ML) estimation, distributed learning.

I. INTRODUCTION

Federated learning (FL) enables machine learning models to be trained in a distributed way[1][2]. In an FL setting, a federation of edge devices, such as Internet of Things devices, mobile phones, etc., trains a shared global model collaboratively under the coordination of a central server while keeping all the training data locally at the edges. This alleviates the growing privacy concern of sharing the data and reduces the communication cost caused by sending extensive data to a central node.

A typical federated learning system with a star network topology, illustrated in Fig. 1, consists of a central server and many clients, where each client holds a local dataset to train a shared global model in a coordinated manner iteratively. At each training iteration, each client trains the model with its local dataset and then uploads the updated parameters to the server. The central server combines all the uploaded parameters from all the servers by algorithms such as Federated Averaging (FA)[1], which simply takes average of the received parameters from different clients, and then broadcasts the combined parameters to all the clients. A client will then update the local model with the received parameters, which completes one iteration of training.

![Figure 1. A typical federated learning system with federated averaging algorithm[1].](image-url)

Constantly updating the model parameters, which could be tens of millions for a deep neural network like ResNet[3], causes a massive communication overload. This could significantly slow down the convergence of FL as the communication links between the devices and the central server is usually with limited bandwidth and power[4]. Therefore, many research efforts have been made to theoretically characterize the fundamental trade-offs between the performance of the trained models and the communication costs of the FL framework[1][2][3][4][5]. Meanwhile, from a more practical point of view, other efforts have been made to reduce the total communication cost. Most of them focus on the uplink communication and attempt to either compress the parameters to be uploaded or reduce the frequency of uploading actions[6][7][8][9][10][11][12][13]. Another direction of reducing the uplink communication overhead is to compress the gradient before sending to the server. Many algorithms have been proposed to this end, e.g., gradient sparsification[14], gradient quantization[15][16][17], and sparse ternary compression[18]. Regarding the download communication, some existing methods also compressing the broadcast message from the server to the clients[19].
These works have remarkably eased the communication burden of the FL systems, however, at the cost of degrading the model performance. This has motivated researchers to consider the models inherently with low precision parameters for the sake of less communication cost to upload these parameters and preserve the model performance at the same time [10]. The first attempt is made by J. Xu et al., who have proposed Federated Trained Ternary Quantization (FTTQ) in [20], where ternary quantization is applied on all parameters of the model. Inspired by this work, we investigate the federated learning of the binary neural networks (BNN), a type of neural network with only binary parameters, in this paper.

BNN was first proposed by I. Hubara et al. in 2016 [21], where they show the BNN achieves nearly the same performance as the corresponding neural networks with real-valued parameters, which has brought a lot of research attention since. During the training of a BNN, a set of auxiliary real-valued parameters is used to keep the accumulation of gradients and update the binary parameters accordingly. A trivial way of training BNN in an FL setting is for each client to upload the auxiliary real-valued parameters to the server to aggregate and then feedback the aggregated parameters to the clients. However, this results in the exact communication cost of training a real-valued neural network, which is against the initial motivation of employing BNN. To reduce the communication cost, it is preferable to send only the binary parameters to the server, which may degrade the performance of the BNN as the auxiliary real-valued parameters may not be correctly updated at each iteration of training. In this paper, we propose a novel federated learning algorithm of training the BNNs, referred to as BiFL-BiML, that only uploads the binary parameters to the server, and employs a novel maximum likelihood-based parameter updating approach, referred to as ML-PU, to estimate the aggregated auxiliary parameters with the local auxiliary parameters and the received aggregation of binary parameters from all the clients. The main contributions of this paper are summarized as follows:

- We propose a novel federated learning framework of training BNN, which uploads only binary values instead of 32-bits float points to the server such that the communication cost of the FL system is drastically reduced. A novel maximum likelihood-based approach is proposed to estimate the aggregated auxiliary parameters and update the local auxiliary parameters accordingly, which preserves the performance of the BNN training even when only the binary parameters are aggregated during the training.

- To the best of our knowledge, we are the first to theoretically investigate the convergence of a general centralized BNN and derive a set of conditions under which BNNs are guaranteed to converge towards the optimum. We further extend the convergence analysis to the BNN under FL settings considered in this paper.

- Numerical results validate the effectiveness of the proposed method, which show it reduces the upload communication cost from 32 bits to 1 bit per parameters when comparing with the conventional FL implementation that sends the real-valued parameters in the uplink communication, while the performance loss incurred by the binarization can be further compensated by a hybrid method. Moreover, the proposed BiFL-BiML remarkably outperforms the other communication efficient FL implementations that upload the quantized parameters, including FTTQ [20] and other BiFL algorithms which do not utilize ML-PU to update the local auxiliary parameters, demonstrating the effectiveness of ML-PU.

This paper is organized as follows. In Section III, we present the preliminaries about the federated learning and the BNN. In Section III, we present the proposed federated learning framework of training BNN in Section III, followed by the theoretical analysis on the convergence of the proposed method in Section IV. Section V provides the numerical results and Section VI concludes the paper.

II. PRELIMINARY

In this section, we first present the necessary preliminaries about federated learning for this work and then introduce the original version of BNN [21] and the BNN training method we employ in this work.

A. Federated Learning

We show an example of federated systems in Fig. [1], which consists of several clients connected to a central server. The training data is distributed among all the clients either in an IID or a non-IID manner. A typical federated learning algorithm operates iteratively with four phases, i.e., the local training phase, the parameter uploading phase, the parameter aggregating phase and parameter downloading phase.

During the local training phase, each client trains a common model shared by all the clients with its local data and by a particular neural network training algorithm. The parameter uploading phase happens after the local training phase when all the clients transmit the updated parameters to the central server. The communication cost occurring during this phase can be prohibitive as every client in the system needs to send a whole set of all parameters of the model to the server. After receiving all the local parameters from the clients, the server aggregates parameters in a certain way to generate one set of parameters of the model during the parameter aggregating phase, which will then be broadcast to all the clients in the parameter downloading phase. One widely used method to aggregate parameters is Federated Averaging (FA) algorithm [1], which we use as a baseline to compare the novel parameter aggregating method proposed in this paper. We will explain the process of the FA algorithm in details in the sequel. After the parameter downloading phase, each client then initiates the model with the received parameters from the server and starts a new iteration of local training.

Here, we show the details of the Federated Averaging algorithm. Without loss of generality, assume that the considered federated learning system is to solve a typical image classification problem with a labeled dataset \( D \). Note that we overlook the details of communication protocol as it is out of the scope of this paper and assume that all sent
models can be received successfully. Each client caches a non-overlapping subset of the dataset $D_i$, denoted by $D_i$, where $i = 1, \ldots, M$, and $M$ is the number of clients in the system. $D_i$ contains $|D_i|$ image-label pairs denoted as $(x_{ij}, y_{ij})$, where $j = 1, \ldots, |D_i|$, $x_{ij}$ represents the $j$th image at the $i$th client, and $y_{ij}$ denotes the corresponding label. We emphasize that the subsets of the dataset at the clients are non-overlapping, i.e., $D_i \cap D_j = \emptyset, \forall i \neq j$, and hence $|D| = \sum_{i=1}^{M} |D_i|$. We denote the local parameters of the common model at client $i$ as $W^i$, $i = 1, \ldots, M$. The FA algorithm aggregates all the local parameters uploaded by the clients as follows:

$$\hat{W} = \frac{1}{|D|} \sum_{i=1}^{M} |D_i| W^i,$$  \hspace{1cm} (1)

where the derived set of parameters is the weighted sum of $W^i$, $i = 1, \ldots, M$ with the sizes of local datasets as the weights.

Without loss of generality, assume that all the clients employ the same loss function, given as $\ell(y, g(x, W))$, where $y$ is the true label of the image $x$, and $g(x, W)$ denotes the predicted label by the model specified by the set of parameters $W$. Hence we have the average loss of the local dataset at a certain client $i$, denoted by $F_i(W)$, as follows:

$$F_i(W) = \frac{1}{|D_i|} \sum_{j=1}^{D_i} \ell(y_{ij}, g(x_{ij}, W)),$$  \hspace{1cm} (2)

and we then define the global loss as follows:

$$F_g(W) = \frac{1}{|D|} \sum_{i=1}^{M} \sum_{j=1}^{|D_i|} \ell(y_{ij}, g(x_{ij}, W))$$  

$$= \sum_{i=1}^{M} \frac{|D_i|}{|D|} F_i(W),$$  \hspace{1cm} (3)

The communication between the server and the clients takes place during the parameter uploading phases and the parameter downloading phases. The communication cost of the latter can be neglected as the transmission of the global parameters to all the clients can be done in a broadcast manner. Whereas in the parameter uploading phase, each client has to unicast a different set of parameters to the server, which may introduce prohibitive communication costs. Hence in this paper, we focus on the minimization of the communication cost during this phase.

B. Binary Neural Networks

Binary neural network is a type of neural network with highly compressed parameters first introduced in [21], where the neural network weights are binary, requiring much less storage space compared to the classical neural networks.

Consider a $L$-layer binary neural network, where the weights and activations are constrained to $+1$ or $-1$. We denote the weights (including trainable parameters of the activation) of each layer by $W_l^b \in \{1, -1\}^*$, where $l = 1, \ldots, L$ and * represents the dimension of $W_l^b$. We further introduce a scalar parameter $\vartheta_l$ for each layer acting as the trainable amplitude for this layer, which has been shown to improve the performance of BNN in [22]. The output of layer $l$ is given as follows:

$$a_l = \vartheta_l f_l(W_l^b, a_{l-1})$$  \hspace{1cm} (4)

where $a_{l-1}$ is output of layer $l - 1$, and $f_l(\cdot)$ denotes the operation of layer $l$ on the input.

Here we explain how to train a BNN with the stochastic gradient descent (SGD) algorithm. Denote the learning rate as $\eta$. We introduce $\hat{W}$ as the auxiliary real-valued parameters of the same size as $W^b$. We employ the SGD algorithm to update the network parameters as well as $\vartheta$ iteratively. The corresponding parameters at each iteration are specified by the subscript $t$, where $t = 1, \ldots, T$ are the total iterations. At training iteration $t$,

$$W_t \leftarrow W_{t-1} + \eta \frac{\partial \ell(W_{t-1}, \vartheta_{t-1}, D)}{\partial W_{t-1}}$$  \hspace{1cm} (5a)

$$\vartheta_t \leftarrow \vartheta_{t-1} + \eta \frac{\partial \ell(W_{t-1}, \vartheta_{t-1}, D)}{\partial \vartheta_{t-1}}$$  \hspace{1cm} (5b)

where $D$ denotes the training data used in this iteration. $\ell(\cdot)$ is the loss function, the value of which depends on network parameters $W^b$, $\vartheta$ and the current training batch $D_t$. Hence, the second terms on the right hand side of (5) correspond to the gradients on $W^b$ and $\vartheta$, respectively. Then, $W^b$ is being updated at the end of iteration $t$ as follows:

$$W_t^b \leftarrow \text{Sign}(W_t),$$  \hspace{1cm} (6)

where $\text{Sign}(\cdot)$ is an element-wise operation that returns $+1$ if the element is positive and $-1$ otherwise.

We further restrict the values of the auxiliary parameters $\hat{W}_t$ within a given bound such that these parameters would not grow overly large whereas the corresponding binary parameters remain the same. Hence, at the end of the iteration, we have [21]

$$\hat{W}_t \leftarrow \text{clip}(\hat{W}_t, -1, 1)$$  \hspace{1cm} (7)

where $\text{clip}(\cdot, 1, -1)$ is an element-wise operation that returns $1$ for any element larger than $1$, $-1$ for any element smaller than $-1$, or the same value for any element in between.

We emphasize that the auxiliary real-valued parameters $\hat{W}$, are only used during the training to enable iterative optimization of the binary parameters of the BNN as shown above. During inference time, the output of each layer of the BNN is derived by the binary parameters $W^b$ and the scalar parameter $\vartheta$ as given by (4).

III. FEDERATED LEARNING WITH BINARY NEURAL NETWORKS

We consider a federated learning system where each client stores and trains a binary neural network of an identical structure to others’ with local data, referred to as BiFL. In this section, we first present different ways of implementing BiFL. Then for the communication efficient implementation, we propose a novel parameter updating approach that preserves the performance of the BNN as the communication-heavy counterparts.
Algorithm 1: BiFL algorithm

1. Initialize: All the clients load a neural network of the same architecture locally, and initialize the parameters of this neural network with the same parameters; each client informs the server the size of local dataset, i.e., $|D_i|$, $i = 1, ..., M$.

2. for $i = 1$ to $T$ do

   3. Start a global iteration

   4. 1. The local training phase

      for all clients $i = 1$ to $M$ do

      6. All clients train a BNN simultaneously

      7. Calculate the loss of BNN: $\ell(W_{t-1}^{b,i}, \vartheta_{t-1}^{i}, D_i)$

      8. Update local parameters by (5)

      9. Binarization: $W_{t}^{b,i} \leftarrow \tilde{W}_{t}^{i}$

      end for

   10. 2. The parameter uploading phase

   11. Each client upload $\overrightarrow{W}_t$ or $W_{t}^{b,i}$ to the server, $i = 1, ..., M$.

   12. Each client also uploads $\vartheta_{t}^{i}$ to the server, $i = 1, ..., M$.

   13. 3. The parameter aggregating phase

   14. The server obtains the global parameters by applying FA algorithm, i.e.,

   $W_t = \frac{1}{|D|} \sum_{i=1}^{M} |D_i| W_t^{b,i}$ or

   $\tilde{W} = \frac{1}{|D|} \sum_{i=1}^{M} |D_i| \tilde{W}_{t}^{i}$

   15. The server also obtains the global scalar parameters by applying FA algorithm, i.e.,

   $\vartheta_t = \frac{1}{|D|} \sum_{i=1}^{M} |D_i| \vartheta_{t}^{i}$.

   16. 4. The parameter downloading phase

   17. Each client downloads $W_t$ or $\tilde{W}$ from the server, and use the local update method to update $W_{t}^{i}$.

   18. Each client also updates the scalar parameters:

   $\vartheta_{t}^{i} \leftarrow \vartheta_t$.

   19. Binarization: $W_{t}^{b,i} \leftarrow \tilde{W}_{t}^{i}$

A. The Implementation of BiFL

We elaborate in this section how to train a binary neural network in a federated learning setting. The detailed implementation of BiFL is shown in Algorithm 1. Following the process of federated learning, each client first initializes the auxiliary real-valued parameters of the local binary neural network, denoted by $\overrightarrow{W}$, and the scalar parameters for all the layers $\vartheta_t^i$, where $i$ is the index of the client, $i = 1, ..., M$. And then the initial $W_0^{b,i}$ is derived by applying the Sign function on $W_0$, i.e.,

$$W_0^{b,i} \leftarrow \text{Sign}(W_0).$$  (8)

After the initialization, the system starts iterative training. During the local training phase of iteration $t$, $\forall t \in [1 : T]$, each client derives the updated parameters $\overrightarrow{W}_t$, $\vartheta_t^i$, and $W_t^{b,i}$ following the training process described in Section II-B using its local dataset. After the local training, each client attempts to further update the local parameters collaboratively with others clients. One way is to obtain the globally aggregated data from the server by sending the local weights. The aggregated scalar parameters, denoted by $\vartheta_t$, could be obtained by following the same process of traditional federated learning. That is, each client uploads $\vartheta_t^i$, $i = 1, ..., M$ during the parameter uploading phase, and then the server aggregates these parameters utilizing the FA algorithm, i.e.,

$$\hat{\vartheta}_t = \frac{1}{|D|} \sum_{i=1}^{M} |D_i| \vartheta_t^i,$$

where we recall that $D_i$, $i = 1, ..., M$, are the disjoint local datasets at users, and $D$ is the union of all the local datasets. Then the server broadcasts $\hat{\vartheta}_t$ and each client replaces $\vartheta_t^i$ with $\hat{\vartheta}_t$, i.e., $\vartheta_t^i \leftarrow \hat{\vartheta}_t$, $i = 1, ..., M$.

We present in the following different methods of further updating $W_t^{b,i}$ and $\tilde{W}_t$ using global information after local training at each iteration of federated learning and illustrate the difference among them in Fig. 4.

1) The most straightforward way is for each client to upload $W_t$ to the server. Then following the same process of updating $\vartheta_t^i$, the server derives the globally aggregated auxiliary real-valued parameters, denoted by $\tilde{W}_t$, by using the FA algorithm, and then broadcast to all the clients. Each client then updates the local parameters as $\tilde{W}_t \leftarrow \tilde{W}_t$, and $W_t^{b,i} \leftarrow \text{Sign}(\tilde{W}_t)$. We refer to BiFL with this method as BiFL-Full. We note that this way does not take advantage of the binarization of the network parameters, and end up with the same communication cost with typical neural networks.

2) It is more desirable to upload only $W_t^{b,i}$ than $\tilde{W}_t$ for the purpose of communication cost reduction. In this method, the server also aggregates the received $W_t^{b,i}$ by the FA algorithm, that is, $W_t = \frac{1}{|D|} \sum_{i=1}^{M} |D_i| W_t^{b,i}$, where the aggregated parameters $W_t$ is real-valued. Upon receiving $W_t$ from the server, each client updates the local parameters as $\tilde{W}_t \leftarrow \beta \text{Sign}(W_t) + (1 - \beta) \tilde{W}_t$, $W_t^{b,i} \leftarrow \text{Sign}(\tilde{W}_t)$, where $\beta$ is a predetermined parameter that ranges in $[0, 1]$, which governs the significance of $W_t$ on $\tilde{W}_t$. We refer to BiFL with this method as BiFL-Bi-UpOnly.

3) This method, referred to as BiFL-Bi-UpDown, is similar to the method 2) where each client uploads the binary parameters. The only difference is that the server further binarizes the aggregated parameter, and broadcasts $\text{Sign}(W_t)$ instead of $W_t$. And the clients update local parameters by $\tilde{W}_t \leftarrow \beta \text{Sign}(\tilde{W}_t) + (1 - \beta) \tilde{W}_t$, $W_t^{b,i} \leftarrow \beta \text{Sign}(W_t) + (1 - \beta) W_t^{b,i}$, $i = 1, ..., M$. This method further reduces the communication cost compared to the former one. However, since the transmission of the aggregated parameters is in a broadcast manner, the reduction is not significant. Note that when $\beta = 1$, BiFL-Bi-UpDown is equivalent to the algorithm proposed in [20] despite that it works on BNNs but not ternary neural networks.

4) The last method we present here shares the same process as the Method 2) except that a novel maximum likelihood estimation based approach is employed to update $\tilde{W}_t$ using the received $\tilde{W}_t$ and the local parameters $W_t^{b,i}$, which we will elaborate in details in the next section.
We refer to the corresponding BiFL implementation as BiFL-BiML. The numerical results show that this approach achieves better convergence than BiFL-BiUpOnly and BiFL-Bi-UpDown.

**B. Maximum Likelihood Estimation Based Parameter Updating**

In this section, we present the proposed novel maximum likelihood estimation based parameter updating approach, referred to as ML-PU, in details. This approach, employed by BiFL-BiML, aims to use the received $\tilde{W}_t$ from the server and the local parameters $W_i$ to approximate $W_t$, i.e., the globally aggregated auxiliary real-valued parameter by using the FA algorithm. In such way, BiFL-BiML can train the local BNN models to have as good performance as by BiFL-Full while greatly reduces the communication cost. In the following, for the ease of presentation, we omit the subscript $t$ since the process presented below applies to any iteration. For simplicity, we assume that all clients hold the same amount of data, i.e., $|\mathcal{D}_1| = ... = |\mathcal{D}_M|$. We will later show that the proposed approach can be easily generalized to scenarios where the amounts of data on each client are different.

We denote by $\mathcal{W}_j$ and $\hat{\mathcal{W}}_j$ the $j$th parameter within the local set of the auxiliary parameters at user $i$, i.e., $\mathcal{W}_i$, and the corresponding $j$th parameter of the received aggregations $\tilde{W}$, respectively, $j = 1, ..., N$, where $N$ is the total number of parameters in the BNN model. We consider the real-valued auxiliary parameters $\mathcal{W}_i$ on each client as samplings of normally distributed random variables. More specifically, we assume that for any $j \in \{1, ..., N\}$, the $j$th real-valued parameters at all $M$ clients, i.e., $\mathcal{W}_j^1, ..., \mathcal{W}_j^M$, are independent samplings from the same normal distribution $N(\mu_j, \sigma_j^2)$, where $\mu_j$ and $\sigma_j$ is the expectation and standard derivation of the normal distribution. We also assume that all the parameters $\mathcal{W}_i^1, ..., \mathcal{W}_i^M$ at client $i$, $i = 1, ..., M$, are independently distributed.

The process of ML-PU is to estimate $\mu_j$ independently at each client with the local $\mathcal{W}_j$ and the received $\tilde{\mathcal{W}}_j$, and update the local parameter $\mathcal{W}_j$ with this estimation, denoted by $\hat{\mu}_j^i$, at the end of the iteration, i.e., $\mathcal{W}_j^i \leftarrow \hat{\mu}_j^i$. We note that since the local parameters at different clients are usually not identical, the estimations $\hat{\mu}_j^1, ..., \hat{\mu}_j^N$ are therefore of different values. As a result, the updated auxiliary parameters are also not identical across clients, different from the case with BiFL-Full. In the following, we elaborate how to obtain $\hat{\mu}_j^i$ with maximum likelihood estimation. For the ease of presentation, we omit the superscript $i$ and the subscript $j$ since the approach presented next applies to any $i \in \{1, ..., M\}$ and any $j \in \{1, ..., N\}$.

Upon receiving $\hat{\mathcal{W}}$, one can easily calculate the number of clients that have uploaded the binary parameters of value $+1$.
With the knowledge of $M_P$, which yields $M_P = (\bar{w} + 1)M/2$. We denote the number of clients that have uploaded $-1$ by $M_N$, and have that $M_N = M - M_P$. With the knowledge of $M_P$ and $M_N$ and the local auxiliary parameter $\bar{w}$, each client uses the maximum likelihood criterion to estimate the average $\mu$ of the corresponding normal distribution as follows:

$$
\arg \max_{\mu, \sigma} \left( \frac{M - 1}{M_P - 1} \right) \times \left( P \{ x \geq 0 \mid x \sim N(\mu, \sigma) \} \right)^{M_P - 1} \times \left( P \{ x < 0 \mid x \sim N(\mu, \sigma) \} \right)^{M_N - 1} f(\bar{w}, \mu, \sigma),
$$

where $I$ denotes a indicator that equals to 1 if the argument inside the brackets is true and 0 otherwise. $f(\cdot, \mu, \sigma)$ represents the probability density function of normal distribution $N(\mu, \sigma^2)$, i.e. $f(\bar{w}, \mu, \sigma) = (2\pi\sigma^2)^{-1/2}e^{-(w-\mu)^2/2\sigma^2}$.

We note that since $x \sim N(\mu, \sigma)$, $(x-\mu)/\sigma \sim N(0, 1)$. We denote by $z(a)$ the upper $a$ quantile of the standard normal distribution, i.e. $z(a) = \int_a^{\infty} f(t, 0, 1)dt$. We have

$$
P \{ x \geq 0 \mid x \sim N(\mu, \sigma) \} = P \left\{ \frac{x - \mu}{\sigma} \geq -\frac{\bar{w} - \mu}{\sigma} \right\} \sim N(0, 1) \right\} = z(\frac{-\mu}{\sigma}) = 1 - z(\frac{\mu}{\sigma})
$$

We then apply the logarithmic transformation and the above transformation to (9) and remove terms that are irrelevant of $\mu$ and $\sigma$. The maximization problem then becomes

$$
\arg \max_{\mu, \sigma} \left( M_P - I(\bar{w} > 0) \right) \ln(1 - z(\frac{\mu}{\sigma})) + (M_N - I(\bar{w} < 0)) \ln(z(\frac{\mu}{\sigma})) - \ln \sigma - \frac{(\bar{w} - \mu)^2}{2\sigma^2}.
$$

We then add the term $1/2 \ln \bar{w}^2$ to the objective function, and replace $\mu/\sigma$ by $u$ and $\bar{w}/\sigma$ by $v$, respectively, to simplify (11). It yields

$$
\arg \max_{u, v} \left( M_P - I(\bar{w} > 0) \right) \ln(1 - z(u)) + (M_N - I(\bar{w} < 0)) \ln(z(u)) + \frac{1}{2} \ln v^2 - \frac{(v - u)^2}{2},
$$

$$
s.t. \quad v \geq 0 \right\}
$$

which is equivalent to the optimization problem given in (11). The constraint is due to the fact that $v = \bar{w}^2/\sigma$ which should not be negative.

We take derivative of the objective function with regards to $v$, and obtain an analytical solution of it for the optimization problem, which is

$$
v = \frac{u + \text{Sign}(\bar{w})\sqrt{u^2 + 4}}{2},
$$

Replace $v$ with the derived analytical solution. The optimization problem (12) becomes

$$
\hat{u} \triangleq \arg \max_u \ln f(u) = (M_P - I(\bar{w} > 0)) \ln(1 - z(u)) + (M_N - I(\bar{w} < 0)) \ln(z(u))
$$

$$
+ \ln \frac{\sqrt{u^2 + 4} + u \text{Sign}(\bar{w})}{2} - \frac{(u + \text{Sign}(\bar{w})\sqrt{u^2 + 4} - u)^2}{2}.
$$

Given the values of $M$, $M_P$ and the sign of $\bar{w}$, we can solve the above optimization problem and thus obtain $\hat{u}$ numerically.

Note that for the considered FL system, the number of clients, i.e., $M$, is fixed throughout the whole training process, whereas $M_P$ is usually different for each iteration. To speed up the estimation of $\mu$ during training, we aim to obtain the approximate analytic relation between $\hat{u}$ and $M_P$, denoted by $h(\cdot)$, such that $\hat{u} \approx h(M_P, M, \bar{w})$. Noting that $f(u)$ contains term Sign($\bar{w}$), we consider the two cases, i.e., $\bar{w} > 0$ and $\bar{w} < 0$, separately. For a given $M$, we first derive $\hat{u} - M_P$ curves for both cases by numerically solving (14), and then derive $h(\cdot)$ by fitting these two curves. For instance, we derive the $\hat{u} - M_P$ curves for $M = 100$ which are presented in Fig 3 (a) and (b), for $\bar{w} > 0$ and $\bar{w} < 0$, respectively. We can easily observe that the curves are approximately logarithmic thus we fit the curves with the following analytic function

$$
h(M_P, M = 100, \bar{w}) = \begin{cases} 
\frac{a_1 + a_2 \ln(M_P + a_3)}{a_4 + a_5 \ln(-M_P + a_6)} & \text{if } \bar{w} > 0 \\
\frac{a_4 + a_5 \ln(-M_P + a_6)}{a_1 + a_2 \ln(M_P + a_3)} & \text{if } \bar{w} < 0 
\end{cases}
$$

where $a_1, a_2$ and $a_3$ are parameters to be optimized to fit the $\bar{w} > 0$ curve, while $a_4, a_5$ and $a_6$ for the $\bar{w} < 0$ curve. From the symmetry of (14), we know that $a_4 = -a_1$, $a_5 = -a_2$ and $a_6 = M + a_3$, so we only needs to fit one curve. We also present the curves derived by the fitted analytic functions, for $\bar{w} > 0$ and $\bar{w} < 0$, in Fig 3 (a) and (b), respectively, where we can observe that the difference between the curves derived numerically and the approximation curves is negligible.
According to (13), we have the estimation of \( v \) given by
\[
\hat{v} = \hat{u} + \text{Sign}(\hat{\mu})\sqrt{\hat{u}^2 + 4}.
\]
(16)
Recalling that \( u = \mu/\sigma \) and \( v = \hat{\mu}/\sigma \), we have
\[
\hat{\mu} = \hat{u}\hat{\sigma}/\hat{v} = \frac{(\text{Sign}(\hat{\mu})\sqrt{\hat{u}^2 + 4} - \hat{u})\hat{\sigma}}{2}.
\]
(17)
As we can see in Fig. 3 (c) and (d), \( \frac{\text{Sign}(\hat{\mu})\sqrt{\hat{u}^2 + 4} - \hat{u})\hat{\sigma}}{2} \), i.e., \( \hat{\mu}/\hat{\sigma} \) takes value in range \((-1, 1)\) for a large range of \( M_p \), which means using the estimation presented above ends up with \(|\hat{\mu}| < |\hat{\sigma}|\). This also implies that \( \hat{\mu} \) is not an unbiased estimation of \( \mu \) as \( \mu \) is the average value of all \( \hat{\mu} \) at clients. Hence by updating \( \hat{\mu} \) with \( \hat{\sigma} \) at the end of each iteration, \( \hat{\mu} \) may be converging to zero, and lead to the failure of training. To tackle the issue raised above, we introduce a hyperparameter \( \alpha \) to our proposed algorithm and update local real-valued parameters by
\[
\overline{\mu} \leftarrow \text{clip}(\alpha \hat{\mu}, -1, 1),
\]
(18)
where \( \alpha \) is a constant larger than 1. We note that as auxiliary parameters do not take part in the forward computation, the scaling of them has no impact on the network output. We also note that the auxiliary real-valued parameters during the training of a BNN model are accumulations of gradients scaled by the chosen learning rate. The impact of this scaling on the convergence will be discussed in Section IV-A and how to determine the value of \( \alpha \) in Section IV-B.

Recall that we assumed each client has the same amount of data at local. Below, we extend the proposed ML-PU method to a more general scenario where the sizes of datasets are not identical across clients. To apply the proposed method, for each client \( i \), we assume a virtual FL system with \( M_i = |D|/|D_i| \) clients, where each client has a dataset of the \( |D_i| \). With the received \( \tilde{W}_k \), client \( i \) therefore can update the local auxiliary parameters by the ML-PU method with \( M = M_i, M_p = (\hat{w} + 1)M_i/2. \) We note that \( M_i \) may not be an integer, which, however, is still valid to derive the update of the auxiliary parameters following the steps of ML-PU. We validate the effectiveness of this approach with numerical results presented in Section V.

IV. Theoretical Analysis

In this section, we investigate the conditions under which the parameters of binary models will converge towards the optimal, and also discuss the impact of hyperparameter \( \alpha \) on the convergence.

A. The Convergence of Binary Models

Here we prove the convergence of gradient descent algorithm used to train binary models. We first consider a model with real-valued parameters \( \overline{W} \), where each element of \( \overline{W} \) is clipped between 1 and \(-1\). We denote by \( F(W) \) as the average loss on the dataset \( D \) by this model given the set of parameters \( W \), i.e. \( F(W) = \frac{1}{|D|} \sum_{i=1}^{|D|} \ell(y_i, g(x_i, W)) \). Recall that \( \ell(y_i, g(x_i, W)) \) is the loss between the true label, denoted by \( y_i \), and the prediction, given by \( g(x_i, W) \), of the input sample \( x_i \).

Following the literature on the convergence of gradient based training algorithms, we make the following assumptions, which have been proven to be satisfied for squared-SVM and linear regression models [23].

Assumption 1. \( F(W) \) satisfies

1) \( F(W) \) is \( \xi \)-strong convex and first-order derivable, i.e.,
\[
F(W_1) - F(W_2) \leq \nabla F(W_1)^T(W_1 - W_2) - \frac{\xi}{2}\|W_1 - W_2\|^2
\]
for any \( W_1 \) and \( W_2 \).

2) \( F(W) \) is \( \rho \)-Lipschitz, i.e. \( \| F(W_1) - F(W_2) \| \leq \rho \| W_1 - W_2 \| \) for any \( W_1 \) and \( W_2 \).

3) \( F(W) \) is \( \beta \)-smooth, i.e., \( \| \nabla F(W_1) - \nabla F(W_2) \| \leq \beta \| W_1 - W_2 \| \) for any \( W_1 \) and \( W_2 \),

where \( \| \cdot \| \) denotes the 1-2 norm of a vector.

We have the following lemma, referred to as Descent lemma, followed from Lemma 1.2.3 of [24], the proof of which can also be found in [24].

Lemma 1 (descent lemma). If \( F(W) \) satisfies Assumption 1, then
\[
F(W_1) - F(W_2) \leq \nabla F(W_1)^T(W_1 - W_2) - \frac{1}{2\beta}\|\nabla F(W_1) - \nabla F(W_2)\|^2.
\]

Denote by \( W^b \) the product of applying sign function element-wisely to \( \overline{W} \), i.e., a binarized version of \( \overline{W} \), such that \( W^b \in \{-1, 1\}^N \). We have Lemma 2 which specifies the math relation between \( W^b \), \( \overline{W} \) and any \( V^b \neq W^b, V^b \in \{-1, 1\}^N \).

Lemma 2. For \( W^b \) and \( \overline{W} \) defined as above and any \( V^b \neq W^b, V^b \in \{-1, 1\}^N \), if there is \( K \) non identical elements among all the \( N \) dimensions in \( W^b \) and \( V^b \), we have the following two conclusions:

1) \( \| W^b - V^b \| \leq 2\| \overline{W} - V^b \| \); 
2) \( \| W^b - V^b, \overline{W} - V^b \| \leq \arccos \sqrt{K/N} \),

where \( \langle \cdot, \cdot \rangle \) refers to the angle between two vectors.

Proof. The proof of the lemma can be found in Appendix A.

We then prove the convergence of \( W^b \) towards the optimal values, which are denoted by \( W^* \), i.e.,
\[
W^* \triangleq \arg \min_{W^b \in \{-1, 1\}^N} F(W^b)
\]
and by \( \epsilon = \| \nabla F(W^*) \| \) the norm of the gradient at \( W^* \). We denote by \( W^b_t \) the parameters of the binary model at time \( t \), \( \overline{W}_t \) the corresponding auxiliary real-valued parameters, and \( K_t \) the number of nonidentical elements in \( W^b_t \) and \( \overline{W}_t \), then
\[
\| W^b_t - W^* \| = 2\sqrt{K_t}.
\]
(19)
Recall that the auxiliary parameters in the binary model are updated by \( \overline{W}_{t+1} = \overline{W}_t - \eta \nabla F(W^b_t) \) at time \( t \). Lemma 3 below gives a sufficient condition under which the auxiliary parameters converge to the optimal set of binary parameters \( W^* \).
Lemma 3. For a binary model described above, if the loss function $F(\cdot)$ meets Assumption 1 and the learning rate $\eta$ satisfies

$$0 < \eta < \phi(\lambda_t, K_t),$$

where $\lambda_t \triangleq \|\nabla F(W_t^b)\|/\epsilon$, $\phi(\lambda_t, K_t) = \frac{\lambda_t - 1}{\sqrt{\beta^3 \lambda_t^2 N (\lambda_t^2 + 1)}} (\sqrt{\lambda_t^2 (N - K_t)} - \sqrt{\lambda_t^2 (N - K_t)} - \sqrt{\lambda_t^2 (N - K_t)} - \sqrt{\lambda_t^2 (N - K_t)}),$ and $\beta$ and $\xi$ are defined in Assumption 1, then we have

1) $\frac{K_0 \epsilon}{\lambda_t} \leq \lambda_t \leq \frac{K_0 \beta + \epsilon}{\lambda_t},$

2) $\|W_{t+1} - W^*\|^2 < \|W_t - W^*\|^2 - 4(\phi(\lambda_t, K_t) - \eta)\|\nabla F(W_t)\|.$

Proof. The proof of the lemma can be found in Appendix B.

We then discuss the conditions given in Lemma 3 and convergence rate achieved by the gradient descent algorithm for binary models. Note that $\phi(\lambda_t, K_t)$ can be rewritten as

$$\phi(\lambda_t, K_t) = \frac{\lambda_t - 1}{\lambda_t} \cos(\arccos \sqrt{\frac{\xi}{\beta}} + \arccos \sqrt{\frac{K_t}{N}} + \arccos \lambda_t \sqrt{\lambda_t^2 + 1}).$$

(20)

We can easily observe that $\phi(\lambda_t, K_t)$ increases with both $\lambda_t$ and $K_t$. Hence, there exists a pair of $(\lambda_0, K_0)$ where $\lambda_0 = K_0 \epsilon/\lambda_t - 1$, such that $\phi(\lambda_t, K_0) > \eta$, for any $\lambda_t > \lambda_0$ and $K_t > K_0$. This satisfies the condition in Lemma 3 on $\phi(\lambda_t, K_t)$.

Hence, with Lemma 3, we can conclude that the auxiliary parameters can always converge to at least a suboptimal set of binary parameters if the loss function $F(\cdot)$ meets the conditions given in Assumption 1. Recall that $\epsilon \triangleq \|\nabla F(W_t^*)\|$, which is close to zero since $W^*$ is the optimal set of parameters. Also note that $\lambda_t \triangleq \|\nabla F(W_t)\|/\epsilon$, hence $\lambda_t \rightarrow \infty$, which yields $\phi(\lambda_t, K_t) \approx \cos(\arccos \sqrt{\frac{\xi}{\beta}} + \arccos \sqrt{\frac{K_t}{N}}).$ Note that when $K_t \approx \frac{(\beta - \xi)N}{\beta}$, we have $\phi(\lambda_t, K_t) \geq 0.$ Hence, we can guarantee the model to converge to have at most $\frac{(\beta - \xi)N}{\beta}$ different elements to the optimal solution. That is, $\frac{(\beta - \xi)N}{\beta}$ is an upper bound on difference between the trained model and the optimal solution. We have the following theorem showing the convergence rate of binary models by the gradient descent algorithm.

Theorem 1. The gradient descent algorithm for binary models provides geometric convergence to a set of parameters $W^b$ that are with at most $K_0$ elements different from the optimized binary parameters $W^*$, that is, $\forall \kappa > 0$, there exists a minimum $T_0$ such that for $t \geq T_0$

$$K_t = \|W_t^b - W^*\| < 2 < K_0 + \kappa,$$

and $T_0 = O(\log \frac{1}{K_0 + \kappa})$ for any initial $W_0$.

Proof. The proof of the theorem can be found in Appendix C.

We then discuss the impact of scaling the ML estimation with hyperparameter $\alpha$ to update the weights as shown in on the convergence of BiFL-BiML. We note that this is equivalent to scaling the learning rate. Therefore, similarly, if $\alpha$ is too small, the model will be too slow to converge. On the other hand, when $\alpha$ is too big, the model may converge too quickly to a suboptimal solution. Hence we normally choose a value of $\alpha$ that is slightly larger than 1.

B. The Convergence of Binary Models in FL Setting

In this section, we discuss the convergence of BiFL-BiML, and explore the favorable range of the scaling parameter $\alpha$. We have shown that the ML estimation is not unbiased in Section III-B. Recall that the introduction of $\alpha$ is to scale the estimation to be unbiased. The following theorem shows that BiFL-BiML converges if each parameter $\overline{\omega}$ is updated with an unbiased estimation in every iteration.

Theorem 2. If each parameter $\overline{\omega}$ is updated with an unbiased estimation and Assumption 1 holds, BiFL-BiML is of geometric convergence in time and the number of clients $M$, that is, $\forall \kappa > 0$, there exists a minimum $T_0$ such that for $t \geq T_0$

$$\|\overline{\omega}_{m,t} - W^*\|^2 < \frac{K_0 + \kappa}{2},$$

and $T_0 = O(\log \frac{1}{K_0 + \kappa})$ for any initial $\overline{W}_{m,0}$.

Proof. The proof of the theorem can be found in Appendix D.

Note that it is intractable to obtain a close-form expression on the value of $\alpha$ that imposes unbiased estimation. As aforementioned in Section IV-A, $\alpha$ is preferably slightly larger than 1, and numerical results reveals that the range of $\alpha$ leads to a satisfactory performance of the proposed algorithm is between 1.25 and 2.

V. Numerical Results

To numerically evaluate the proposed federated learning framework with BNN, We consider a simple neural network shown in Fig. 4 where we denote by “Conv $m \times n \times n$” a $m$-channel convolutional layer with a $n \times n$ kernel, “FC $m$” a fully connected layer with output size of $m$, “BN” a batch normalization layer, “$\ell_2$” a pooling layer downsampling the input by a factor of 2, and both “TanH” and “Softmax” the hyperbolic tangent and softmax activation function, respectively. We note that the last layer uses the softmax activation instead of tanh to get the predicted labels. We evaluate the performance of the proposed method on three datasets. The first is the handwritten digit recognition dataset MNIST, the second is the Federated Extended MNIST (FEMNIST) dataset, which is a well-known dataset for federated learning, and the last is the CelebA, which is a face recognition dataset based on the photographs of celebrities, and also a well-known dataset for federated learning. Since MNIST dataset is simpler, we use the smaller group of parameters in Fig. 4 for MNIST and the larger for FEMNIST. We use a ResNet with 4 convolutional layers and a full-connect layer for CelebA. The details of our neural network can be found in our code.

We compare the proposed model to other federated learning algorithms in terms of both the accuracy and the communication cost.

https://github.com/yuzhiyang123/FL-BNN
A. Settings

We consider a star-topology federated learning system with one server and \( M \) clients. We distribute our dataset to each of the clients. The algorithms we compare to are listed in the following, as well as the ways of distributing datasets to the clients.

1) Compared algorithms: We compare the following algorithms numerically in this section, where we note that all of them employ the same neural network structure, the same learning rate, and are trained with the same optimizer. The neural network of BiFL is exactly the same as that of baseline and FA, except that it applies binarization on convolutional and full-connected layers.

- **Baseline:** the centralized learning algorithm, where the whole dataset is located on a single computing center, and the neural network are with real-valued parameters.
- **FA:** the federated learning algorithm proposed in [1], where the neural network are with real-valued parameters. The dataset is distributed among all the clients.
- **BNN:** the centralized learning algorithm as the baseline, but the neural network is with binary parameters instead, trained in the way as explained in Section II-B.
- **FTTQ:** the algorithm raised in [20], which is a communication efficient FL algorithm for ternary neural networks.
- **BiFL-Full, BiFL-Bi-UpOnly, BiFL-Bi-UpDown and BiFL-BiML:** which are different implements of federated learning system with BNN, which we explain in Section III in details.

2) Dataset: We use two datasets in the experiments. The first dataset we use is the MNIST dataset [25], which consists of 70000 28 \( \times \) 28 gray-scale handwritten images. We divide it into a training set of 60000 images and a testing set of 10000 images, where all images of either set are evenly distributed in 10 classes. The second dataset we use is the Federated Extended MNIST (FEMNIST) dataset [26], which is a well-known dataset for federated learning. The FEMNIST dataset is based on the Extended MNIST dataset, which is a more complicated image classification problem. The FEMNIST dataset is based on the FEMNIST dataset, which is a more complicated image classification problem. More specifically, the dataset contains images of handwritten numbers and English alphabets, which has 62 classes in total. It is recommended to use only a part of it to validate FL algorithms [26], since the dataset is very large, and the distributed algorithms will not be well examined if the learning ability of the clients is too high. In our experiments, we randomly select 10% of the Extended MNIST dataset for each experiment. For similar reason, we randomly select 50% of the CelebA dataset for each experiment. Reference [26] also provides code for partitioning the FEMNIST and CelebA dataset by both IID and non-IID manners, and for evaluating methods. We evaluate our algorithm following their work.

3) Data distributions: We consider three way of distributing data to the clients: IID, non-IID and unbalanced data size, which we will explain in the following.

- **IID data:** For MNIST, the dataset is evenly distributed to the clients with regards to the categories, i.e., all clients have the same amount of images from each class. For FEMNIST, we use the IID partition in [26]. We only reserve 500 images for each client to ensure that all clients have the same amount of data. We finally get 89 clients with 500 images on each client and a testing dataset of 5500 images. To partition the CelebA dataset, we use the same way with FEMNIST, and get 100 clients with 500 images on each client and a testing dataset of 5000 images.
- **non-IID data:** For MNIST, we divide the data from each class into \( MN_c/10 \) subdatasets. Then we distributed a random set of \( N_c \) among the total \( MN_c \) subdatasets to each client. We note that, in the case, each client ends up with no more than \( N_c \) classes of images in the local dataset, and the total amount of images on each client is identical. For FEMNIST, we use the non-IID partition in [26]. Like in IID data, we only reserve 500 images for each client to ensure that all clients have the same amount of data. We finally get 89 clients with 500 images on each client and a testing dataset of 5500 images. To partition the CelebA dataset, we also use the same way with FEMNIST, and get 100 clients with 500 images on each client and a testing dataset of 5000 images.
- **Unbalanced data size:** For MNIST, clients have different sizes of local datasets, while the amount of images from each category at each client is the same, where 20 clients among 100 evenly share 40% of the dataset, another 40 share another 40% of the dataset, whereas the remaining 40 share the remaining 20% of the dataset. For FEMNIST, we firstly divide the dataset into 178 subdatasets in an IID manner. We then distribute them on 89 clients in three categories similarly, and the clients in each category get 1, 2, and 4 subdatasets respectively. The amounts of clients in each category are 36, 35, and 18.

4) Basic configuration: We use the following configuration in our experiments.

- **Total amount of clients:** \( M = 100 \) for MNIST and \( M = 89 \) for FEMNIST by default.
- **Learning rate:** Decaying learning rate, \{0.005, 0.002, 0.001\} for MNIST, \{0.0003, 0.0001, 0.00005\} for FEMNIST, and \{0.1, 0.05, 0.02\} for CelebA. The learning rate is decayed after the 30th and 60th epochs, respectively.
- **Gradient descent algorithm:** Adam [27], a widely used gradient based optimizer.
- **Batch size:** \( B = 64 \).
- **Hyperparameter \( \alpha \) in BiFL-BiML:** 1.25 for MNIST and 1.75 for FEMNIST, since they leads to the best performance in terms of accuracy.
B. Performance with IID Data

In this section, we compare all the algorithms and evaluate the effect of the scaling parameter $\alpha$ on the IID data distribution.

1) Performance comparison of different algorithms: We compare the performance of all different BiFL implements in terms of the accuracy and also the four benchmarks introduced above in Fig. 5. In Fig. 5, the shadows showing the standard deviation scaled by 1/5. We note that the training processes of FL and BiFL-Full are not shown completely in the figures of communication costs. All the algorithms are evaluated on the same IID distributed datasets. We set $\beta = 0$ for both BiFL-Bi-UpOnly and BiFL-Bi-UpDown, as this values leads to the best performance of both algorithms in terms of accuracy. We also include the numerical result of BiFL-Bi-UpDown with $\beta = 1$ as it is a straightforward extension of the algorithm proposed in [20]. We also list the accuracy and the required communication costs by different algorithms for both uploading and downloading in Table II. All the data is averaged over 5 independently initialized dataset partitions. Note that the required downloading cost per parameter of our algorithm is only $\log (M + 1)$ bit due to the fact that there are only $M + 1$ values of the aggregated parameters as there are $M$ clients, which may increase if the dataset is not IID or there are more clients, but no more than 32 bit.

We can see that the centralized algorithms, i.e., the baseline and BNN, converges the fastest and results in the better accuracy than the federated learning algorithms as expected, however, there is a certain loss caused by binarization. We can also find a certain gap on the accuracy between all the distributed algorithms and corresponding centralized algorithms. It can be observed that the performance of our BiFL-BiML algorithm is not too worse than the performance of FA and BiFL-Full and exceeds the other three distributed algorithms, i.e., BiFL-Bi-UpOnly, FTTQ, and BiFL-Bi-UpDown, while reducing the uploading communication cost from 32 bits to 1 bit. We notice that BiFL-BiML also converges faster than the other three communication efficient FL algorithms. We note that BiFL-Bi-UpDown and FTTQ require smaller download communication cost, however, the difference is neglectable due to the broadcasting, especially when the number of clients is big. The better performance of the BiFL-BiML algorithm in terms of accuracy demonstrates the significance of updating the local model with high precision aggregated parameters. Since there is still a certain gap in accuracy between BiFL-BiML and FL, we can use a simple hybrid method to overcome this, which will be discussed later.

2) Effect of the scaling parameter: We compare the performance in terms of accuracy when different values of scaling parameter $\alpha$ are used on MNIST, as shown in Fig. 6. The result demonstrates that with a larger $\alpha$, our algorithm results in a lower accuracy, which validates our analysis. It also demonstrates that our algorithm may not converge to a good result when $\alpha$ is very small.

3) Performance on hybrid algorithm: Although our BiFL-BiML algorithm performs well in terms of accuracy comparing with other quantization schemes, there is still a notable performance gap comparing with that of FA, which is more significant with the FEMNIST dataset. To achieve a closer performance to FA and maintain a low communication cost at the same time, we train the NN in a hybrid way, in particular, by BiFL-BiML for the initial $T$ epochs, and by FA for the rest. We show the results on FEMNIST when $T = \{20, 40, 60\}$ in
From Fig. 7, we can observe that the hybrid algorithm can achieve the performance of FA with lower communication cost. We also note that the test accuracy drops significantly at time \( T \) when switching the algorithm, but converges to the training curve of FA in a short time. This is due to the forward pass switching from using the binary parameters to using the real value parameter at time \( T \). Due to the difference in the datasets on different clients, the models may also differ at the beginning, which is similar to the initialization of FA. It can also be seen that a smaller \( T \) results in faster convergence as expected with slightly larger communication cost for the same test accuracy.

### C. Effect of the Number of Clients

We compare the result derived by the proposed algorithm and the FA algorithm when the number of clients in the system is different. In particular, we set the number of clients to be half, the same, and double of that of the IID settings. We present the smoothed results in Fig. 8. We can observe from Fig. 8 that the performance difference caused by the different amounts of clients is insignificant for all algorithms.

### D. Performance with Non-IID Data

We compare different algorithms when the data is distributed to the clients in a non-IID manner as described above shown in Table II. All the results are derived by 5 independent experiments with independently initialized dataset partitions. We recall that the \( N_c \) is the largest number of data classes each client can have according to our data partition method. From Table II, it can be observed that the proposed BiFL-BiML algorithm outperform the FTTQ and the BiFL-Bi-UpOnly algorithms in terms of accuracy even on non-IID data, whereas a remarkable performance gap to that of FA still presents.

### E. Effect of Unbalanced Data Size

We evaluate the performance of the distributed algorithms when the datasets at clients are of different sizes. In particular, the data is allocated to each client in the way that is described above. As shown in Fig. 9, we can find that the performance gap between the our BiFL-BiML algorithm and the FA algorithm is still small but more notable than the case with identical dataset size. Also note that BiFL-BiML algorithm still significantly outperform the FTTQ and BiFL-Bi-UpOnly algorithms.

### F. Effect of Partial Participation

Here, we evaluate the performance of the distributed algorithms when not all the clients upload at each iteration. We
denote the participation rate of the system as \( \lambda \), i.e., only \( \lambda M \) clients upload their parameters at each time slot. We show the results for both IID and non-IID data distributions in Fig. [10] For MNIST, non-IID case, we set \( N_c = 3 \). As we can see in Fig. [10] from the results on MNIST, the participation rate of clients has marginal impact on the performance in terms of accuracy when using IID data distributions, whereas causing a notable drop in performance when \( \lambda \) is small and using non-IID data distributions. From the results on FEMNIST, we can find that only the BiFL-BiML has obvious performance drop. This means that the data distribution of FEMNIST is not too far away from the IID data distribution, and that the BiFL-BiML is more sensible of partial participation mainly because the ML-PU is very sensible of the amount of the uploaded sets of parameters.

### VI. Conclusions and Future Directions

In this paper, we proposed a novel federated learning algorithm, BiFL-BiML, to train neural networks with binary parameters. This algorithm introduces significant reduction in communication cost as the client uploads binary parameters instead of 32-bit float point parameters as in conventional federated learning systems. It employs a maximum likelihood estimation based parameter updating approach, ML-PU, in which each client estimates the auxiliary real-valued parameters based on the binary models’ aggregation result received from the server and then updates its local parameters. To the best of our knowledge, we are the first to have theoretically derived the conditions on the convergence of training binary models by gradient descent algorithm. We numerically compared the proposed BiFL-BiML with the conventional FL algorithm, a communication-efficient FL algorithm for ternary neural works called FTTQ [20], and two trivial implementations of FL learning with BNN, i.e., BiFL-Bi-UpOnly and BiFL-Bi-UpDown, respectively, which utilize simpler parameter updating mechanisms instead of ML-PU. The numerical results revealed that BiFL-BiML achieves performance close to the conventional FL algorithm while providing drastic reduction of communication cost, and significantly outperforms FTTQ, BiFL-Bi-UpOnly, and BiFL-Bi-UpDown under all the IID data, non-IID data, and unequal dataset size scenarios. The numerical results also revealed that the neural network can achieve nearly the same accuracy with a reduction of communication cost by a hybrid implementation of BiFL-BiML and the conventional FL algorithms.

### Appendix

#### A. Proof of Lemma 2

We first prove 1). Denote \( \overline{w}_i \), \( w^b_i \) and \( v^b_i \) as the \( i \)th element of \( \overline{W} \), \( W^b \) and \( V^b \), respectively. Define \( I \triangleq \{ i | w^b_i \neq v^b_i, 0 < i \leq N \} \). \( \overline{I} \) has \( K \) elements in total. Recall that the real-valued parameters \( \overline{W} \) are clipped between -1 and 1, i.e. \(-1 \leq \overline{w}_i \leq 1 \). We can find that for any \( i \in I \), \( w^b_i \) and \( \overline{w}_i \) have the same sign but the opposite to \( v^b_i \). Therefore, we have \( 1/2 |w^b_i - v^b_i| = 1 - |\overline{w}_i - v^b_i| < 2 \). We also have for any \( i \notin I \), \( 1/2 |w^b_i - v^b_i| = 0 \leq |\overline{w}_i - v^b_i| \). Combine both cases and it yields \( 1/2 \|W^b - V^b\|_2 \leq \|\overline{W} - V^b\|_2 \), which completes the proof of conclusion 1) in Lemma 2.

We prove 2) next. We have

\[
(W^b - V^b)^T (\overline{W} - V^b) = \sum_{i=1}^{N} (w^b_i - v^b_i)(\overline{w}_i - v^b_i) \geq \sum_{i \in I} (w^b_i - v^b_i)(\overline{w}_i - v^b_i)
\]

(21)

Notice that \( (w^b_i - v^b_i)(\overline{w}_i - v^b_i) > 0 \). Hence, we can apply Cauchy-Schwarz inequality, which yields

\[
(W^b - V^b)^T (\overline{W} - V^b) \geq \sqrt{\sum_{i \in I} (w^b_i - v^b_i)^2 \sum_{i \in I} (\overline{w}_i - v^b_i)^2}
\]

(22)

Also notice that

\[
|\overline{w}_i - v^b_i| = |w^b_i - v^b_i| - |\overline{w}_i - w^b_i| = 2 - |\overline{w}_i - w^b_i| \geq 1, \forall i \in I
\]

(23)

and

\[
|\overline{w}_i - v^b_i| = |\overline{w}_i - w^b_i| \leq 1, \forall i \notin I.
\]

(24)

We have

\[
\frac{N}{K} \sum_{i \in I} (\overline{w}_i - v^b_i)^2 \geq \sum_{i \in I} (\overline{w}_i - v^b_i)^2 + \frac{N - K}{K} \sum_{i \in \overline{I}} (\overline{w}_i - v^b_i)^2 \geq \sum_{i \in I} (\overline{w}_i - v^b_i)^2 + N - K
\]

(25)
Bring (27) to (23), it yields

\[
(W^b - V^b)^T (W - V^b) \geq \sqrt{\frac{K}{N}} \|W^b - V^b\|^2 \|W - V^b\|^2,
\]

which completes the proof of Lemma 2.

B. Proof of Lemma 3

By applying Lemma 1, and we have

\[
F(W_t^b) - F(W^*) \leq \nabla F(W_t^b)^T (W_t^b - W^*) - \frac{1}{\lambda_t} \|\nabla F(W_t^b) - \nabla F(W^*)\|^2,
\]

(27)

\[
F(W^*) - F(W_t^b) \leq \nabla F(W^*)^T (W^* - W_t^b) - \frac{1}{\lambda_t} \|\nabla F(W^*) - \nabla F(W_t^b)\|^2.
\]

(28)

Adding (27) and (28), it yields

\[
(\nabla F(W_t^b) - \nabla F(W^*))^T (W_t^b - W^*) \geq \frac{1}{\beta} \|\nabla F(W_t^b) - \nabla F(W^*)\|^2.
\]

(29)

Since $F(\cdot)$ has $\xi$-strong convexity, we can obtain

\[
F(W_t^b) - F(W^*) \leq \nabla F(W_t^b)^T (W_t^b - W^*) - \frac{\xi}{2} \|W_t^b - W^*\|^2,
\]

(30)

\[
F(W^*) - F(W_t^b) \leq \nabla F(W^*)^T (W^* - W_t^b) - \frac{\xi}{2} \|W^* - W_t^b\|^2,
\]

(31)

and, hence,

\[
(\nabla F(W_t^b) - \nabla F(W^*))^T (W_t^b - W^*) \geq \xi \|W_t^b - W_t^b\|^2.
\]

(32)

By combining (29) and (32), we have

\[
(\nabla F(W_t^b) - \nabla F(W^*))^T (W_t^b - W^*) \geq \sqrt{\frac{\xi}{\beta} \|\nabla F(W_t^b) - \nabla F(W^*)\|^2 \|W_t^b - W^*\|^2},
\]

(33)

since the right side term of (33) is the geometric mean of those of (29) and (32). Note that $\cos(\alpha, \beta) = \frac{a^T b}{|a||b|}$, we can rewrite (33) as

\[
|\langle \nabla F(W_t^b) - \nabla F(W^*), W_t^b - W^* \rangle| \leq \arccos \sqrt{\frac{\xi}{\beta}}.
\]

(34)

By applying Lemma 2 and (34), we have

\[
|\langle \nabla F(W_t^b) - \nabla F(W^*), W_t^b - W^* \rangle| \leq |\langle \nabla F(W_t^b) - \nabla F(W^*), W_t^b - W^* \rangle|
\]

\[
+ |\langle W_t^b - W^*, W_t^b - W^* \rangle|
\]

\[
\leq \arccos \sqrt{\frac{\xi}{\beta}} + \arccos \sqrt{\frac{K}{N}}.
\]

Note that $\|\nabla F(W_t^b)\| = \lambda_t \|\nabla F(W^*)\|$. Hence,

\[
|\langle \nabla F(W_t^b), \nabla F(W^*) - \nabla F(W_t^b) \rangle| \leq \arctan \frac{1}{\lambda_t}
\]

\[
+ \frac{\lambda_t - 1}{\lambda_t} \|\nabla F(W^*)\| \leq \|\nabla F(W_t^b) - \nabla F(W^*)\| \leq \frac{\lambda_t + 1}{\lambda_t} \|\nabla F(W_t^b)\|.
\]

(36)

By combining (35) and (36), it yields

\[
|\langle \nabla F(W_t^b), W_t^b - W^* \rangle| \leq |\langle \nabla F(W_t^b), W_t^b - W^* \rangle|
\]

\[
+ |\langle \nabla F(W_t^b), \nabla F(W^*) - \nabla F(W^*) \rangle|
\]

\[
= \arccos \sqrt{\frac{\xi}{\beta}} + \arccos \sqrt{\frac{K}{N}} + \arccos \frac{\lambda_t}{\lambda_t^2 + 1}
\]

\[
= \arccos \frac{\lambda_t}{\lambda_t - 1} \phi(\lambda_t, K_t).
\]

(37)

From the definition of $\beta$-smooth, the property of $\phi$-strong convex, and (36), we can easily get

\[
\frac{\xi \lambda_t}{\lambda_t + 1} \leq \|W_t^b - W^*\| \leq \frac{\beta \lambda_t}{\lambda_t - 1},
\]

(38)

which can be easily transformed to the conclusion 1) in Lemma 3.

Now, recall that $\|W_t^b - W^*\| = 2\sqrt{K}$, the distance between $W_{t+1}$ and $W^*$ is given as follows

\[
\|W_{t+1} - W^*\|^2
\]

\[
= \|W_t - \eta \nabla F(W_t^b) - W^*\|^2
\]

\[
\leq \|W_t - W^*\|^2 + \eta^2 \|\nabla F(W_t^b)\|^2
\]

\[
- 2 \eta \nabla F(W_t^b)^T (W_t - W^*)
\]

\[
= \|W_t - W^*\|^2 + \eta^2 \|\nabla F(W_t^b)^2\|^2 - 2 \eta \|\nabla F(W_t^b)\|^2
\]

\[
\leq \|W_t - W^*\|^2 \cos(\nabla F(W_t^b), W_t - W^*)
\]

\[
\leq \|W_t - W^*\|^2 + \eta^2 \|\nabla F(W_t^b)\|^2
\]

\[
- 2 \phi(\lambda_t, K_t) \eta \beta \lambda_t
\]

\[
\lambda_t - 1 \|\nabla F(W_t^b)^2\|^2 \|W_t^b - W^*\|^2.
\]

(39)

\[
\leq \|W_t - W^*\|^2 + \eta^2 \|\nabla F(W_t^b)\|^2
\]

\[
- \phi(\lambda_t, K_t) \eta \beta \lambda_t
\]

\[
\lambda_t - 1 \|\nabla F(W_t^b)^2\|^2 \|W_t^b - W^*\|^2.
\]

\[
\leq \|W_t - W^*\|^2 - 4 \phi(\lambda_t, K_t) - \eta \beta \lambda_t
\]

\[
\lambda_t - 1 \|\nabla F(W_t^b)^2\|^2 \|W_t^b - W^*\|^2
\]

\[
< \|W_t - W^*\|^2 - 4 \phi(\lambda_t, K_t) - \eta \beta \lambda_t.
\]

\[
< \|W_t - W^*\|^2 - 4 \phi(\lambda_t, K_t) - \eta \beta \lambda_t K_t.
\]

where $a$ is derived from (35) and the definition of $\delta$, $b$ is derived by applying Lemma 2, and $c$ comes from (38). The proof of conclusion 2) in lemma 3 is completed.

Thus, the proof of Lemma 3 has been completed.
\[ C. \text{ Proof of Theorem 1} \]

Recall that \( K_t \) is the amount of different elements in \( W_t \) and \( W^* \). Obviously, when \( K_t \geq K_0 + \kappa \),
\[
\| \mathbf{W}_{t} - W^* \|^2 < K_t \ast 2^2 + (N - K_t) \ast 1^2 < (3 + N/K_0)K_t. \tag{40}
\]
From Lemma 3, the maximum and minimum value of \( W_\ell \) right hand cannot be negative.

D. Proof of Theorem 2

From Theorem 1, we have proved that the gradient descent for binary models provides geometric convergence, which is the same with real-valued neural networks in the overparameterized setting with constant step size defined in [28]. The only difference is that the model can only converge to a set of suboptimal parameters in the binary case. Since we have the assumption that each parameter \( \mathbf{w} \) is updated with an unbiased estimation, the expectation will not be changed during the parameter aggregating. Hence, from Theorem 5 in [28], we have
\[
EF(W_{m,T}) - F(W^*) \leq O(\beta \exp(-\frac{\mu}{\beta}MT) \cdot \| W_{m,0} - W^* \|^2),
\]
when we choose a reasonable learning rate and Lemma 3 holds. Here, \( F(W) \) is the total loss function of FL, and \( F_\ell(W) = \frac{1}{|D|} \sum_{j=1}^{d_j} \ell(y_{ij}, g(x_{ij}, W)) \). \( \ell(\cdot) \) is \( \beta_0 \)-smooth and \( F(\cdot) \) is \( \beta \)-smooth. \( \nu_{\text{max}} \) and \( \nu_{\text{min}} \) are the maximum and minimum value of \( M \| D \| \), respectively. \( E \) is the number of epochs conducted among each aggregating, hence \( E = 1 \) here.

From the definition of \( \xi \)-strong convexity, we can easily know that
\[
\frac{\xi}{2} \| W_{m,T} - W^* \|^2 - \| W_{m,T} - W^* \| \\
\leq \frac{\xi}{2} \| W_{m,T} - W^* \|^2 + \nabla F(W^*)^T (W_{m,T} - W^*) \tag{43}
\]
\[
\leq \| F(W_{m,T}) - F(W^*) \| \leq O(\text{exp MT}).
\]
Since \( \epsilon \) is a small positive value, we have \( \| W_{m,T} - W^* \| \leq O(\text{exp MT}) \) when Lemma 3 holds. Hence, with \( t_0 = O(\frac{1}{M} \log \frac{1}{K_0 + \kappa}) \), either \( \| W_{b,M,T} - W^* \|^2/2 < K_0 + \kappa \) or Lemma 3 does not hold at a time before \( t_0 \). Both cases indicate that Theorem 2 holds.

REFERENCES

[1] H. B. McMahan, E. Moore, D. Ramage, et al., “Federated learning of deep networks using model averaging,” in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, Fort Lauderdale, FL, USA, Apr. 2017.

[2] J. Konečný, B. McMahan, and D. Ramage, “Federated optimization: Distributed optimization beyond the datacenter,” *arXiv preprint arXiv:1511.03575*, Nov. 2015.

[3] K. He, X. Zhang, S. Ren, et al., “Deep residual learning for image recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, Jun. 2016.

[4] P. Kairouz, H. McMahan, B. Avent, et al., “Advances and open problems in federated learning,” *arXiv preprint arXiv:1912.04977*, Dec. 2019.

[5] Y. Zhang, J. Duchi, M. I. Jordan, et al., “Information-theoretic lower bounds for distributed statistical estimation with communication constraints,” in *Advances in Neural Information Processing Systems (NIPS)*, Lake Tahoe, NV, USA, Dec. 2013.

[6] M. Braverman, A. Garg, T. Ma, et al., “Communication lower bounds for statistical estimation problems via a distributed data processing inequality,” in *ACM Symposium on Theory of Computing (STOC)*, Cambridge, MA, USA, Jun. 2016.

[7] Y. Han, A. Özgür, and T. Weissman, “Geometric lower bounds for distributed parameter estimation under communication constraints,” in *Conference on Learning Theory (COLT)*, Stockholm, Sweden, Jul. 2018.

[8] J. Acharya, C. L. Canonne, and H. Tyagi, “Inference under information constraints: lower bounds from chi-square contraction,” *IEEE Transactions on Information Theory*, vol. 66, no. 12, pp. 7835–7855, Dec. 2020.

[9] L. P. Barnes, Y. Han, and A. Özgür, “Learning distributions from their samples under communication constraints,” *arXiv preprint arXiv:1902.02890*, Feb. 2019.

[10] J. Konečný, H. B. McMahan, F. X. Yu, et al., “Federated learning: Strategies for improving communication efficiency,” *arXiv preprint arXiv:1610.05492*, Oct. 2016.

[11] A. T. Suresh, F. X. Yu, S. Kumar, et al., “Distributed mean estimation with limited communication,” in *International Conference on Machine Learning (ICML)*, Sydney, Australia, Aug. 2017.

[12] J. Konečný and P. Richtárik, “Randomized distributed mean estimation: Accuracy vs communication,” *Frontiers in applied mathematics and statistics*, vol. 4, pp. 2297–4687, Dec. 2018.

[13] D. Alistarh, D. Grubic, J. Li, et al., “QSGD: Communication-efficient SGD via gradient quantization and encoding,” in *Advances in Neural Information Processing Systems (NIPS)*, Long Beach, CA, USA, Dec. 2017.

[14] S. Horvath, C.-Y. Ho, L. Horvath, et al., “Natural compression for distributed deep learning,” *arXiv preprint arXiv:1905.10988*, May 2019.
[15] A. F. Aji and K. Heafield, “Sparse communication for distributed gradient descent,” in Conference on Empirical Methods in Natural Language Processing (EMNLP), Copenhagen, Denmark, Sep. 2017.

[16] M. Courbariaux, Y. Bengio, and J.-P. David, “Binaryconnect: Training deep neural networks with binary weights during propagations,” in Advances in Neural Information Processing Systems (NIPS), Montreal, QC, Canada, Dec. 2015.

[17] F. Li and B. Liu, “Ternary weight networks,” arXiv preprint arXiv:1605.04711, May 2016.

[18] F. Sattler, S. Wiedemann, K.-R. Müller, et al., “Robust and communication-efficient federated learning from non-iid data,” arXiv preprint arXiv:1903.02891, Mar. 2019.

[19] A. Khaled and P. Richtárik, “Gradient descent with compressed iterates,” arXiv preprint arXiv:1909.04716, Sep. 2019.

[20] J. Xu, W. Du, Y. Jin, et al., “Ternary compression for communication-efficient federated learning,” IEEE Transactions on Neural Networks and Learning Systems (Early Access), pp. 1–15, Dec. 2020. DOI: 10.1109/TNNLS.2020.3041185.

[21] I. Hubara, M. Courbariaux, D. Soudry, et al., “Binarized neural networks,” in Advances in Neural Information Processing Systems (NIPS), Barcelona, Spain, Dec. 2016.

[22] M. Rastegari, V. Ordonez, J. Redmon, et al., “XNOR-Net: ImageNet classification using binary convolutional neural networks,” arXiv preprint arXiv:1603.05279, Mar. 2016.

[23] S. Wang, T. Tuor, T. Salonidis, et al., “Adaptive federated learning in resource constrained edge computing systems,” IEEE Journal on Selected Areas in Communications, vol. 37, no. 6, pp. 1205–1221, Jun. 2019.

[24] Y. Nesterov, “Nonlinear optimization,” in Introductory Lectures on Convex Optimization: A Basic Course, Boston, MA: Springer US, 2004, pp. 1–50, ISBN: 978-1-4419-8853-9. DOI: 10.1007/978-1-4419-8853-9_1.

[25] Y. Lecun, L. Bottou, Y. Bengio, et al., “Gradient-based learning applied to document recognition,” Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[26] S. Caldas, S. M. K. Duddu, P. Wu, et al., “LEAF: A benchmark for federated settings,” arXiv preprint arXiv:1812.01097, Dec. 2018.

[27] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in International Conference on Learning Representations (ICLR), San Diego, CA, USA, May 2015.

[28] Z. Qu, K. Lin, J. Kalagnanam, et al., “Federated learning’s blessing: FedAvg has linear speedup,” arXiv preprint arXiv:2007.05690, Jul. 2020.