

# EPACE-FL: Efficient Privacy-Preserving And Communication in Edge Computing of Federated Learning

Chao-Hsien Hsieh

✉✉✉✉✉

Peidong Wang

wangpeidong611@qq.com

✉✉✉✉✉

---

## Article

**Keywords:** Federated Learning, Privacy-Preserving, Dynamic Quantization, ChaotiMasking, Edge computing

**Posted Date:** October 9th, 2026

**DOI:** <https://doi.org/10.21203/rs.3.rs-11024637/v1>

**License:** © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.  
[Read Full License](#)

**Additional Declarations:** No competing interests reported.

---

# EPACE-FL: Efficient Privacy-Preserving And Communication in Edge Computing of Federated Learning

Chao-Hsien Hsieh<sup>1</sup> • Peidong Wang<sup>2</sup>

<sup>1</sup> School of Engineering, Xi'an International University, Xi'an, China

<sup>2</sup> School of Cyber Science and Security, Qufu Normal University, Qufu, China

✉ Peidong Wang

wangpeidong611.qq.com

## Abstract

With the development of the digital age, data privacy leakage becomes increasingly serious. Achieving efficient distributed training while protecting sensitive data has therefore become an important challenge. Federated learning provides an effective solution by keeping data on local devices and training models collaboratively. However, conventional federated learning still suffers from insufficient privacy protection and low communication efficiency. Furthermore, edge devices usually possess limited computational resources and unstable network bandwidth, which makes deploying heavy machine learning models directly on edge devices extremely difficult. To address these limitations, this paper proposes the EPACE-FL framework which is an efficient privacy-preserving and communication federated learning framework for edge computing. To specifically address the resource limitations and bandwidth fluctuations of edge computing, this framework further introduces Dynamic Quantization and Chaotic Masking (DQCM) algorithm for joint compression and encryption. The EPACE-FL framework consists of FedAvg system-initialization algorithm and DQCM algorithm. During training of federated learning, each client employs FedAvg for initialization to generate its model parameters. To prevent privacy leakage and low communication efficiency when uploading local parameters through unstable edge computing, this paper adopts DQCM algorithm to process parameters in three steps. First, the algorithm dynamically compresses local parameters into shorter integers to adapt to the limited bandwidth of edge computing. Then, each client generates a random chaotic mask to encrypt the integers, thereby enhancing transmission security. Finally, the parameters are securely aggregated on the server. In this way, the problems of inadequate privacy security and low communication efficiency in edge environments are resolved. Experimental results show that, compared with existing mainstream baselines, EPACE-FL framework reduces the total communication overhead per client by 29.3%. When facing malicious privacy attacks, the protection success rate of EPACE-FL framework improves by 41.6% over the baseline algorithm DPFedAvg, fundamentally blocking the reverse reconstruction of private information.

**Keywords** Federated Learning·Privacy-Preserving·Dynamic Quantization·ChaotiMasking·Edge computing

## 1 Introduction

With the rapid development of 5G/6G networks and the Industrial Internet of Things, Edge AI and Industry 4.0 [23,29] have attracted extensive attention. Devices at the network edge, such as smart sensors, autonomous vehicles, and wearable monitors for medical purposes, generate massive amounts of sensitive data. The conventional mechanism of centralized machine learning, which uploads these data to a cloud server, generates extreme pressure on communication and severe risks to privacy. Federated learning [1]-[3] offers a promising paradigm for intelligence at the edge. It enables clients to optimize a global model collaboratively while keeping raw data strictly local.

However, deploying large-scale models of deep learning over realistic networks at the edge introduces two intertwined challenges. communication restricted by dynamic factors and overhead constrained by the privacy protection. Figure 1 illustrates the specific workflow and the model of threats. On one hand, edge computing show feature highly unstable wireless channels, asymmetric uplinks, and dynamic fluctuations in bandwidth. Requiring a massive array of clients to transmit model parameters of high dimensions to the central server frequently leads to congestion of the network and intolerable latency in communication, as illustrated by Number 1 in Figure 1. This latency severely paralyzes the global progress of training [24]. On the other hand, although federated learning avoids the sharing of raw data, a malicious client within the network can intercept the streams of transmission over vulnerable wireless links. This attacker uses the intercepted model parameters to reconstruct private information precisely via attacks that invert

model parameters, as depicted by the red arrow marked as Number 2 in Figure 1 [10, 11].

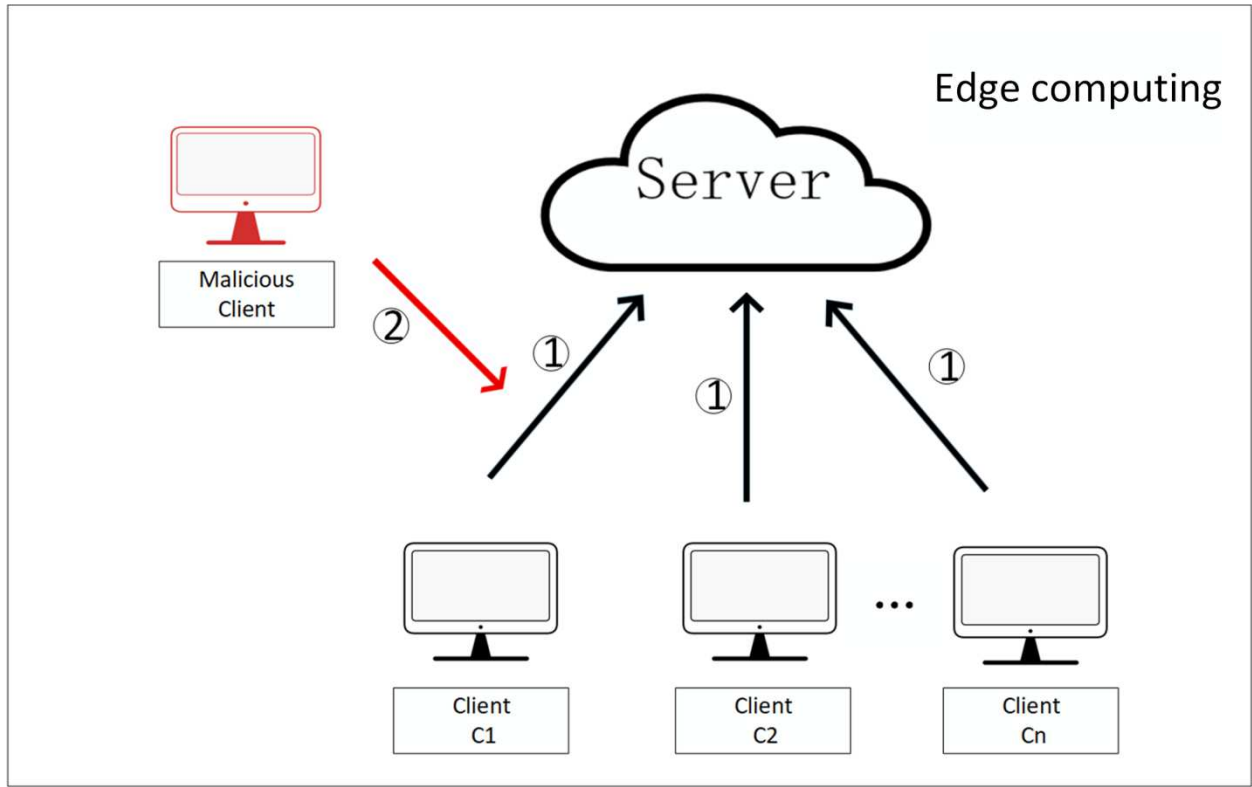


**Fig. 1** Problems Faced by Federated Learning

For example, thousands of wearable medical devices collect the real-time physiological data of patients to train a model for disease diagnosis in a healthcare scenario at the edge computing. Strict privacy-related regulations prohibit raw-data uploads. These regulations include the General Data Protection Regulation. However, applying conventional federated learning over unreliable cellular networks causes frequent timeouts in transmission. Moreover, as developers integrate complex cryptographic protocols, these edge device processors encounter performance failures. This failure causes the latency of computation to exceed acceptable thresholds.

Many studies attempt to address above, but most designs target stable environments centralized in the cloud and ignore the severe limitations in hardware and latency at the edge. Sparsification [1] and Knowledge Distillation [6] reduce the payload but cause the loss of critical

features in complex models. Static Quantization [5] compresses parameters with fixed widths of bits. This static mechanism fails completely to adapt to the unpredictable drops in bandwidth typical of wireless networks at the edge. To protect privacy, current studies introduce differential privacy [12], secure multiparty computation [15]-[18], and fully homomorphic encryption [20]. However, these cryptographic techniques typically induce exponential growth of ciphertexts and require substantial computational resources. These demands make the techniques fundamentally incompatible with edge devices that rely on batteries and possess limited computational resource.

All of these techniques directly transmit plaintext parameters. Recent studies show that attackers can leverage model inference attacks to extract hidden data information [10]. Researchers have therefore introduced differential privacy [12], homomorphic encryption [13,14], and secure multiparty computation [15]-[18] into federated learning. The secure aggregation (SecAgg) [19] and fully homomorphic encryption (FHE) [20] protect data with cryptographic techniques. However, both techniques could compute expensive and then greatly increase data volume, which limits efficient training on edge devices.

This paper focuses on achieving extremely low communication overhead and high-accuracy model aggregation in the premise of privacy preservation for computing at the edge. It therefore proposes EPACE-FL framework that combines chaotic masking and dynamic quantization.

(1) This paper designs EPACE-FL framework specifically tailored for edge computing environments. It effectively accommodates the hardware resource constraints of edge nodes and addresses insufficient privacy protection alongside communication inefficiency.

(2) This paper proposes a DQCM algorithm. It masks quantized parameters with chaotic noise, greatly reduces communication bandwidth, and blocks privacy inference attacks on model parameters.

(3) This paper evaluates the effectiveness of EPACE-FL framework. It verifies the total communication overhead and high-accuracy model aggregation in the premise of privacy preservation.

The remainder of this paper is organized as follows. Section 2 reviews the related work and analyzes the limitations of existing methods. In the Section 3, it introduces the preliminary knowledge, including the standard federated learning paradigm, cryptographic principles, and mathematical models of edge resource constraints. Section 4 details the proposed EPACE-FL framework and the core Dynamic Quantization and Chaotic Masking (DQCM) algorithm. In the Section 5, it presents the experimental setup and comprehensively evaluates the system performance. Finally, Section 6 concludes the paper and discusses future research directions.

## **2 Related Work**

This paper review the related work from four aspects, federated learning within networks at the edge, protection of privacy, efficiency in communication, and dynamic compression of models. This paper analyzes the limitations of existing methods to highlight the unique architectural advantages of the proposed EPACE-FL framework in resource-constrained environments.

### **2.1 Federated Learning for Edge Computing**

Unlike traditional computing in the cloud, edge computing moves the training of models to the periphery of the network. There, devices suffer from highly volatile wireless connections, limited life of batteries, and heterogeneous computational resources. Recent studies [4, 24] emphasize that bottlenecks in communication over unstable uplinks act as the primary cause of slow training. Existing frameworks assisted by the edge primarily focus on the selection of clients

or the allocation of resources to mitigate latency. However, they generally assume a trusted environment and transmit parameters in plaintext. When confronted with malicious eavesdropping on vulnerable links, these methods provide zero resistance. The EPACE-FL framework addresses networks at the edge intrinsically by minimizing local workloads of computation concurrently and adapting transmission payloads dynamically to real-time fluctuations in bandwidth.

## **2.2 Privacy-Preserving Federated Learning**

Although federated learning does not directly transmit raw client data, gradient inversion attacks can still exploit shared model parameters to steal sensitive information. Existing studies mainly adopt methods such as differential privacy and homomorphic encryption. For example, ALDP-FL [21] dynamically adjusts the clipping threshold to reduce the effect of differential-privacy noise on accuracy. Zhu et al. [22] study dynamic privacy-budget allocation to balance model utility. And FedMPS [23] separates shared and private parameters to reduce information leakage.

These privacy-preserving algorithms have strong theoretical foundations, but they usually cause substantial system overhead. Differential privacy continuously adds noise and then reduces final model accuracy, while homomorphic encryption requires extensive computation and increases local storage and communication costs. In contrast, EPACE-FL framework leverages lightweight chaotic masking to protect privacy and reduce additional system overhead.

## **2.3 Adaptive Model Quantization within Dynamic Bandwidth**

When many clients participate or bandwidth is limited, frequent parameter transmission causes high communication overhead. Liu et al. [24] propose adaptive gradient quantization algorithm to fit the communication requirement of heterogeneous devices. Qin et al. [25] exploit dimensionality reduction algorithm to compress model parameters and then reduce the cost of

subsequent homomorphic encryption.

The advantage of such algorithms lies in directly compressing the communication volume at the numerical level; however, they suffer from insufficient adaptability to dynamic network environments and a tendency to lose critical features. In contrast, this paper proposes the EPACE-FL framework, which automatically adjusts the degree of model compression according to the highly volatile network status typical of edge computing environments. When the edge computing bandwidth is severely limited, it compresses data to reduce transmission overhead. As the network condition improves, it preserves complete model accuracy. At the same time, the framework integrates compression and encryption to reduce communication overhead.

#### **2.4 Joint Compression and Secure Aggregation Methods**

To reduce the large data volume and low communication efficiency in federated learning, existing studies commonly adopt model quantization and sparse compression. Some methods also combine privacy protection with compression. For example, Nguyen et al. [26] propose two-level bit-width scheduling to optimize communication volume and privacy risk, while FedMPS [23] removes redundant model parameters through sparsification.

However, existing algorithms exhibit significant shortcomings. most of them decouple model compression and privacy protection into two independent steps. failing to achieve synergistic optimization. As compression is performed before encryption, the compressed plaintext model parameters still carry the risk of information leakage. If encryption precedes compression, the random nature of ciphertext severely undermines the compression efficiency, making it ineffective in reducing communication overhead. To address this issue, this paper designs DQCM algorithm within the PACE-FL framework, which synergistically performs dynamic quantization compression and privacy mask protection. The algorithm directly generates privacy masks on the

quantized compressed parameters, thereby not only circumventing the security risks of plaintext parameter exposure but also fully retaining the communication-optimization benefits of model compression. To compare with traditional stepwise processing approaches, it achieves dual improvements in both efficiency and security.

### 3 Preliminary Knowledge

This section introduces the basic concepts of federated learning, model quantization, and secure masking to support the description of EPACE-FL framework.

#### 3.1 Federated Learning Formulation and Threat Model

The federated learning trains a model with one aggregation server and multiple clients, where the client set is  $\{C = C_1, C_2, \dots, C_N\}$ , client  $C_i$  holds local dataset  $D_i$ , and  $n_i$  denotes its sample size. Equation (1) defines the empirical loss of client  $C_i$  on  $D_i$ .

$$F_i(w) = \frac{1}{n} \sum_{\xi \in D_i} \ell(w; \xi) \quad (1)$$

Here,  $w$  denotes the model parameters and  $\ell(w; \xi)$  denotes the loss on sample  $\xi$ . The global objective of federated learning is given in Equation (2).

$$\min_w F(w) = \sum_{i=1}^N \frac{n_i}{n} F_i(w) \quad (2)$$

In global training round  $t$ , the server selects the participating client set according to the sampling ratio, as shown in Equation (3).

$$S_t \subseteq C \quad (3)$$

The server broadcasts the current global model  $w^t$  to the selected clients. Client  $C_i$  uses  $w^t$  as the initial model, performs  $E$  local training epochs on  $D_i$ , and obtains the local model  $w_i^{t,E}$ . The local model update is given in Equation (4).

$$\Delta w_i^t = w_i^{t,E} - w^t \quad (4)$$

The traditional FedAvg algorithm aggregates local model parameters with weights

proportional to client sample sizes, as shown in Equation (5).

$$w^{t+1} = w^t + \sum_{i \in S_t} \frac{n_i}{\sum_{j \in S_t} n_j} \Delta w_i^t \quad (5)$$

Conventional federated learning employs 32-bit floating-point numbers for transmitting complete model parameters. Large model sizes or numerous global training rounds produce frequent exchanges of high-dimensional parameters, leading to considerable communication costs. In addition, local model parameters can carry distinctive features of client data. Both the server and passive network eavesdroppers can reconstruct training data or infer membership by analyzing the uploaded parameters. Hence, this work pursues minimal communication overhead of model parameters and implements concealment for individual-client parameter content.

### 3.2 Dynamic Quantization and Additive Masking

#### • Dynamic Model Quantization

The EPACE-FL framework translates each floating-point client for model gradient into a low-bit integer and achieves lower model-transfer traffic. In round  $t$ ,  $b_t$  denotes the quantization bit width. Equation (6) gives the maximum signed quantization integer.

$$Q_t = 2^{b_t-1} - 1 \quad (6)$$

Let  $R_t > 0$  be the shared clipping range in round  $t$ . The shared quantization scale is defined in Equation (7).

$$S_t = \frac{R_t}{Q_t} \quad (7)$$

The server determines  $R_t$  from a predefined clipping policy, historical aggregation statistics, or public parameters. The server is not necessary any individual plaintext update from the current round. Client  $C_i$  first clips each element of its local update to  $[-R_t, R_t]$  and then quantizes it as shown in Equation (8).

$$q_i^t = Q_{b_t}(\Delta w_i^t; S_t) = \text{clip}(\text{round}(\frac{\text{clip}(\Delta w_i^t, -R_t, R_t)}{S_t}), -Q_t, Q_t) \quad (8)$$

In this equation,  $Q_{b_t}(\cdot)$  denotes the quantization function with bit width  $b_t$ . The  $q_i^t$  denotes the quantized integer update of client  $C_i$ . The  $\text{round}(\cdot)$  denotes rounding and the  $\text{clip}(x, a, b)$  limits  $x$  to  $[a, b]$ , while the EPACE-FL framework applies clipping and quantization element by element. A client can approximately recover the quantized integer with the shared scale, as shown in Equation (9). The  $\widehat{\Delta w}_i^t$  denotes the approximate model update after de-quantization since quantization introduces rounding errors,  $\widehat{\Delta w}_i^t \neq \Delta w_i^t$ .

$$\widehat{\Delta w}_i^t = Q_{b_t}^{-1}(q_i^t; s_t) = s_t q_i^t \quad (9)$$

A lower quantization bit width reduces communication overhead but produce larger quantization error. Fixed quantization cannot adapt to changing network bandwidth. EPACE-FL framework therefore selects the quantization bit width dynamically according to the current network state and the communication budget of each round. The EPACE-FL framework reduces communication overhead by adopting a low-bit width for low-bandwidth scenarios, while utilizing a higher bit width with favorable network conditions to maintain model accuracy.

## ● Additive Masking

Model quantization reduces communication overhead but cannot prevent an attacker from analyzing an uploaded parameter. The quantized result therefore is necessary an additive mask in the integer domain. For a generic quantized integer  $q_i^t$ , client  $C_i$  generates a random mask  $r_i^t$  and computes the masked upload value in Equation (10).

$$\widetilde{q}_i^t = (q_i^t + r_i^t) \bmod p \quad (10)$$

A lower quantization bit width reduces communication volume but produces larger quantization error. Fixed quantization cannot adapt to changing wide-area network bandwidth.

Dynamic quantization adjusts the bit width according to the current network state. This scheme adopts a higher compression ratio for low-bandwidth scenarios and retains superior parameter precision for good network conditions. Quantization alone cannot prevent analysis of model parameters. Therefore, the client adds a mask to the quantized parameters, as shown in Equation (11).

$$\tilde{q}_i^t = (q_i^t + r_i^t) \bmod p \quad (11)$$

Where  $p$  denotes the modulus used for secure aggregation, the masks of all participating clients satisfy  $(\sum_{i \in S_t} r_i^t) \bmod p = 0$ . Then, the server computes the sum of uploaded mask values and acquires Equation (12).

$$\left( \sum_{i \in S_t} \tilde{q}_i^t \right) \bmod p = \left( \sum_{i \in S_t} q_i^t \right) \bmod p \quad (12)$$

The server can therefore compute the aggregate of all client parameters but cannot directly examine an individual plaintext parameter. Equations (11) and (12) describe the general additive-masking principle. In the EPACE-FL framework, the masked objects are the distance-aware weighted integer parameter  $u_i^t$  and the fixed-point integer aggregation weight  $\hat{\rho}_i^t$ .

### 3.3 Distance-Aware Aggregation Mechanism

Low-bit quantization and Non-IID data affect the accuracy and convergence of federated learning. EPACE-FL framework therefore adopts distance-aware aggregation to reduce the effects of quantization error and statistical heterogeneity by adjusting aggregation weights for the distance between each local model and current global model. The client first derives the dequantized parameter from Equation (9) and then reconstructs the quantized local model in Equation (13).

$$\widehat{w}_i^{t,E} = w^t + \widehat{\Delta w}_i^t = w^t + s_t q_i^t \quad (13)$$

$$d_i^t = \frac{\|w_i^{tE} - w^t\|_2}{\|w^t\|_2 + \varepsilon} \quad (14)$$

Then, the normalized model distance of client  $C_i$  is defined as Equation (14). In this equation,  $\varepsilon$  is a small positive constant that prevents a zero denominator and the  $\|\cdot\|_2$  denotes the Euclidean norm. In order to the normalized distance, the client computes the distance adjustment coefficient in Equation (15).

$$a_i^t = \exp(-\gamma d_i^t) \quad (15)$$

When  $\gamma > 0$  is the distance-sensitivity hyperparameter, the value of  $\gamma$  controls the degree of influence of model distance on aggregation weights. When local models exhibit substantial deviations from the current global model, the model distance  $d_i^t$  gradually increases. As the growing distance reduces the adjustment coefficient  $a_i^t$ , the diminished coefficient effectively limits the impact of divergent models on the final aggregation. The client combines the distance coefficient with its sample size to compute the raw weight in Equation (16).

$$\rho_i^t = n_i a_i^t \quad (16)$$

When  $\gamma = 0$ , this paper have  $a_i^t = 1$  and  $\rho_i^t = n_i$ . The distance-aware aggregation degenerates to FedAvg algorithm. The FedAvg algorithm calculates weights based on client sample sizes. To perform subsequent secure aggregation in the integer domain, the real-valued weight  $\rho_i^t$  is converted into a fixed-point integer. This paper defines the conversion precision as  $\mu$ . This precision dictates the scaling factor  $L = 10^\mu$ . The integer aggregation weight is defined in Equation (17).

$$\hat{\rho}_i^t = \text{round}(L\rho_i^t) = \text{round}(10^\mu \rho_i^t) \quad (17)$$

The client applies the integer weight to the quantized parameters and then computes the weighted integer parameters in Equation (18).

$$u_i^t = \hat{\rho}_i^t q_i^t \quad (18)$$

Here,  $\rho_i^t$  is a scalar integer,  $q_i^t$  is a quantized integer vector, and  $u_i^t$  is a weighted integer

271 model parameter vector.

### 272 3.4 Cryptographic Principles of Chaotic Masking and Secure Aggregation

273 EPACE-FL framework applies pairwise chaotic masks to the weighted parameters and the  
274 integer weight in the integer domain. As a result, this approach prevents an server or passive  
275 network eavesdroppers from obtaining an individual client's parameters. To achieve this  
276 protection, any two clients  $C_i$  and  $C_j$  establish a pairwise shared secret key  $K_{i,j} = K_{j,i}$ . In addition,  
277 this key is not public. Consequently, the server and other clients are unable to acquire this key.

#### 278 • Domain-Separated Seed Generation

279 In round  $t$ , clients  $C_i$  and  $C_j$  generate two independent seeds from the pairwise key, the  
280 round index, and the participating client set, as shown in Equation (19).

$$281 \quad \text{seed}_{i,j,h}^t = H(K_{i,j} \parallel t \parallel \text{Enc}(S_t) \parallel h), h \in \{\text{upd}, \text{wgt}\} \quad (19)$$

282 In this equation,  $H(\cdot)$  denotes a cryptographic hash or key-derivation function. The  $\parallel$   
283 denotes byte-string concatenation.  $\text{Enc}(S_t)$  denotes the canonical encoding of the participating  
284 client set. Moreover, the labels  $\text{upd}$  and  $\text{wgt}$  generate the model-parameter mask and the  
285 aggregation-weight mask, respectively. Different domain labels not only keep the parameter masks  
286 independently from the aggregation-weight masks but also prevent the reuse of the same mask  
287 from revealing relations between plaintext values.

#### 288 • Pairwise Mask Generation

289 Clients employ a chaotic pseudo-random sequence generator to generate pairwise masks. As  
290 shown in Equation (20),  $d$  denotes the dimension of the model update vector.

$$\begin{cases} m_{i,j}^t = \text{CMG}_d(\text{seed}_{i,j,\text{upd}}^t) \bmod p \\ r_{i,j}^t = \text{CMG}_1(\text{seed}_{i,j,\text{wgt}}^t) \bmod p \end{cases} \quad (20)$$

CMG<sub>d</sub>(·) represents a chaotic pseudo-random integer vector with output length d and CMG<sub>1</sub>(·) outputs a single chaotic pseudo-random integer. The  $m_{i,j}^t$  is used to mask the weighted model update and  $r_{i,j}^t$  is used to mask the integer aggregation weight. To ensure that the same client pair can generate identical pairwise masks, a fixed client-index order should be adopted. For example,  $m_{i,j}^t$  and  $r_{i,j}^t$  are defined only for client pairs satisfying  $i < j$ .

### ● Local Total-Mask Construction

Each client determines the signs of pairwise masks from client-index order and constructs the total mask for the model update and the total mask for the local weight, as shown in Equation (21). Since any pairwise mask has opposite signs on the two clients, this paper have  $\sum_{i \in S_t} m_i^t = 0$  and  $\sum_{i \in S_t} r_i^t = 0$  in the condition for all participating clients complete uploading.

$$\begin{aligned} m_i^t &= \sum_{j \in S_t, j > i} m_{i,j}^t - \sum_{j \in S_t, j < i} m_{j,i}^t \\ r_i^t &= \sum_{j \in S_t, j > i} r_{i,j}^t - \sum_{j \in S_t, j < i} r_{j,i}^t \end{aligned} \quad (21)$$

### ● Masked Upload

The client  $C_i$  independently masks the weighted integer update and the integer aggregation weight, as shown in Equation (22).

$$\begin{aligned} y_i^t &= (u_i^t + m_i^t) \bmod p \\ z_i^t &= (\hat{\rho}_i^t + r_i^t) \bmod p \end{aligned} \quad (22)$$

Where  $y_i^t$  denotes the weighted integer-updated vector after masking,  $z_i^t$  represents the integer aggregation weight after masking. The client uploads  $(y_i^t, z_i^t)$  to the server.

## 309    ●    Server-Side Secure Aggregation

310        The server receives the masked uploads from all participating clients and sums them modulo  
 311     $p$ , while the opposite signs of pairwise masks make all masks cancel during aggregation as shown  
 312    in Equation (23).

$$\begin{aligned}
 & \left( \sum_{i \in S_t} y_i^t \right) \bmod p = \left( \sum_{i \in S_t} u_i^t \right) \bmod p \\
 & \left( \sum_{i \in S_t} z_i^t \right) \bmod p = \left( \sum_{i \in S_t} \hat{\rho}_i^t \right) \bmod p
 \end{aligned} \tag{23}$$

314        Model parameters sometimes contain negative values, so the server converts each result  
 315    modulo  $p$  back to a signed integer with the centered modular-decoding function in Equation (24).

$$\text{Dec}_p(x) = \begin{cases} x, & 0 \leq x \leq \frac{p}{2} \\ x - p, & \frac{p}{2} \leq x \leq p \end{cases} \tag{24}$$

317        For vectors,  $\text{Dec}_p(\cdot)$  is applied element-wise. The server performs the computations  
 318    according to Equations (25) and (26).

$$U_t = \text{Dec}_p \left[ \left( \sum_{i \in S_t} y_i^t \right) \bmod p \right] \tag{25}$$

$$U_t = \sum_{i \in S_t} u_i^t, A_t = \sum_{i \in S_t} \hat{\rho}_i^t \tag{26}$$

321        The server dequantizes the aggregate with the quantization scale and then renews the global  
 322    model, as shown in Equation (27).

$$w^{t+1} = w^t + s_t \frac{U_t}{A_t} = w^t + s_t \frac{\sum_{i \in S_t} u_i^t}{\sum_{i \in S_t} \hat{\rho}_i^t} \tag{27}$$

324        Equation (27) gives the distance-aware weighted average of all dequantized client parameters  
 325    without revealing any individual plaintext update. To avoid decoding errors caused by modular  
 326    wraparound, the modulus  $p$  must satisfy Equation (28). Here,  $u_{i,k}^t$  denotes the  $k$ -th element of the  
 327    vector  $u_{i,k}^t$ .

$$p > 2\max \left\{ \max_k \left| \sum_{i \in S_t} u_{i,k}^t \right|, \left| \sum_{i \in S_t} \hat{\rho}_i^t \right| \right\} \quad (28)$$

### 3.5 Edge Resource and Communication Delay Model

To mathematically formulate the severe hardware constraints of edge computing environments, this paper introduces three fundamental delay models. Let the symbol  $\omega$  denote the local computation workload. And the symbol  $\varphi$  denote the edge device processor frequency. Then, defines the local computation latency as shown in Equation (29).

$$T_{comp} = \frac{\omega}{\varphi} \quad (29)$$

Furthermore, let the symbol  $\mu$  represent the transmitted data payload size and let the symbol  $\beta$  represent the dynamic network bandwidth. Equation (30) shows the network communication latency. Consequently, Equation (31) calculates the total system processing latency for an edge node.

$$T_{comm} = \frac{\mu}{\beta} \quad (30)$$

$$T_{total} = T_{comp} + T_{comm} \quad (31)$$

In typical edge computing, the frequency  $\varphi$  remains profoundly small and the bandwidth  $\beta$  fluctuates drastically, which causes the total latency to easily exceed acceptable thresholds. Minimizing the workload  $\omega$  and the payload  $\mu$  therefore becomes mandatory.

## 4 EPACE-FL Framework and Algorithms

### 4.1 EPACE-FL Framework

To accomplish efficient decentralized machine learning alongside absolute privacy preservation, the efficient EPACE-FL framework is introduced in this section. This EPACE-FL framework not only resolves inadequate data protection and transmission inefficiency within

conventional federated systems but also is highly suitable for bandwidth-constrained edge computing. The EPACE-FL framework perfectly accommodates edge computing because it actively minimizes computational overhead and communication overhead described in Equation (31). Through completely avoiding computationally expensive cryptographic operations, the EPACE-FL framework drastically reduces computational workload  $\omega$  required within Equation (29). Simultaneously, DQCM algorithm dynamically converts precision floating-point parameters into low-bit integers. This conversion provides two primary benefits. First, this conversion strictly minimizes the transmission payload  $\mu$  required in Equation (30). Second, this conversion enables edge devices to effortlessly execute the compact DQCM algorithm locally. Consequently, this local execution maintains exceptional comprehensive learning efficiency.

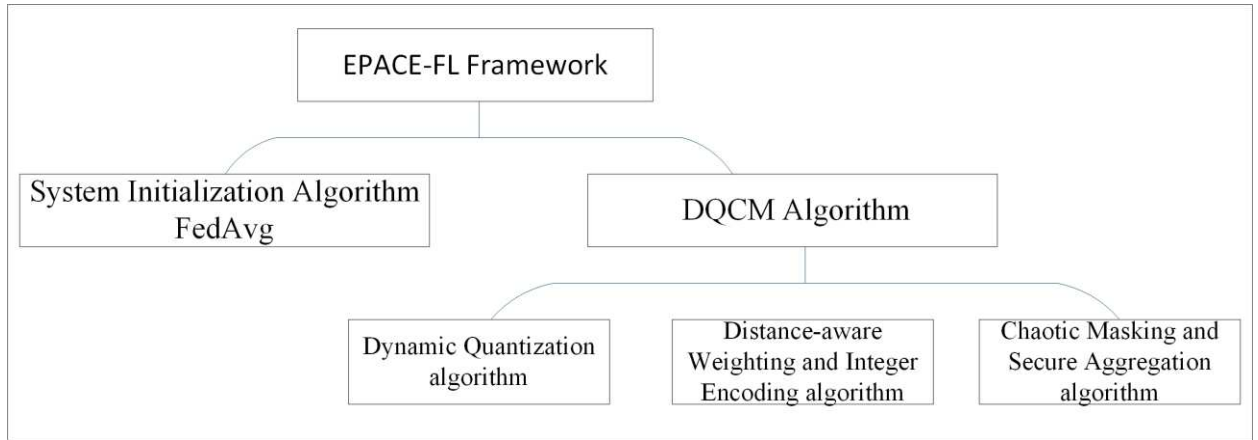


**Fig. 2** PACE-FL Framework

The EPACE-FL framework consists of FedAvg system-initialization algorithm and DQCM algorithm. In each federated-learning round, clients first adopt FedAvg system-initialization algorithm to generate local model parameters. The EPACE-FL framework then adopt DQCM algorithm to compress and encrypt the model parameters, which prevents privacy inference attacks during transmission and reduces network congestion. The server finally performs secure and

366 efficient model aggregation.

## 367 4.2 FedAvg system-initialization (FedAvg) algorithm

368 In EPACE-FL framework, Algorithm 1 uses FedAvg system-initialization algorithm as the  
369 baseline for system initialization and local training. After a client trains on its local dataset and  
370 generates a model update, it sends the update to DQCM algorithm instead of uploading plaintext.

---

**Input:** Edge client datasets  $\{D_i\}_{i=1}^N$ ; initial model  $w^0$ ; communication rounds  $T$ ; sampling ratio  $C$ ; local epochs  $E$ ; learning rate  $\eta$ ; max tolerable edge latency  $T_{max}$ ; edge processor frequency  $f_i$

**Output:** Updated global model  $w^t$ .

- 1: Define the local and global objectives by Eqs. (1) and (2).
  - 2: for  $t = 0, 1, \dots, T - 1$  do
  - 3:     Select the participating client set  $S_t$  by Eq. (3).
  - 4:     Broadcast the global model  $w^t$  to all clients in  $S_t$ .
  - 5:     for edge node  $C_i \in S_t$  in parallel do
  - 6:         //Local training bounded by computation  $T_{comp}$  by Eq. (29)
  - 7:         Train  $w^t$  on  $D_i$  for  $E$  local epochs.
  - 8:         Set  $w^0 \leftarrow w^t$  and train it on  $D_i$  for  $E$  local epochs.
  - 9:         Compute the local update  $\Delta w_i^t$  by Eq. (4).
  - 10:        Retain  $\Delta w_i^t$  locally without plaintext transmission.
  - 11:     end for
  - 12:      $\{q_i^t\}, b_t, S_t \leftarrow \text{DynamicQuantization}(\{\Delta w_i^t\})$  using Algorithm 2.
  - 13:      $\{u_i^t, \bar{a}_i^t\} \leftarrow \text{DistanceWeighting}(\{q_i^t\}, S_t, w^t)$  using Algorithm 3.
  - 14:      $w^{t+1} \leftarrow \text{SecureAggregation}(\{u_i^t, \bar{a}_i^t\}, S_t)$  using Algorithm 4.
  - 15: end for
  - 16: return  $w^t$
-

371

372 **Algorithm 1** FedAvg system-initialization (FedAvg)

373 **4.3 Dynamic Quantization and Chaotic Masking (DQCM) algorithm**

374 To address insufficient privacy protection and low communication efficiency in conventional  
375 federated learning, this paper proposes DQCM algorithm which includes three steps as follows.

376 • **Dynamic quantization compression for communication efficiency**

377 First, with server coordination, the system compresses local model parameters into short  
378 integer sequences. Algorithm 2 selects a shared bit width and scale according to the current  
379 network state and communication budget. Clients convert high-precision floating-point  
380 parameters to low-bit integers. Algorithm 3 then computes distance-aware aggregation weights  
381 and converts them to fixed-point integers. This step reduces the payload and prepares the data for  
382 encryption.

---

**Input:** Local model parameters  $\{\Delta w_i^t\}$ ; network state  $\xi_i$ ; real-time edge computing bandwidth  $B_t$ ; candidate bit widths  $\mathcal{B}$ ; shared clipping bound  $R_t$

**Output:** Quantized model parameters  $\{q_i^t\}$ ; bit width  $b_t$ ; scale  $S_t$ .

- 1: //Evaluate communication latency  $T_{comm}$  constraint by Eq. 30
  - 2: The server selects  $b_t \in \mathcal{B}$  according to  $N_t$  and the round-level communication budget.
  - 3: Compute  $Q_t$  and  $S_t$  by Eqs. (6) and (7).
  - 4: Broadcast  $(b_t, R_t, S_t)$  to all clients in  $S_t$ .
  - 5: for each client  $C_i \in S_t$  in parallel do
  - 6:     Quantize  $\Delta w_i^t$  into  $q_i^t$  by Eq. (8).
  - 7:     Locally reconstruct  $\widehat{\Delta w_i^t}$  by Eq. (9).
  - 8:     Retain  $q_i^t$  locally without uploading it.
-

---

```

9: end for
10: return  $\{q_i^t\}$ ,  $b$ , and  $S_t$ .

```

---

383 **Algorithm 2** Bandwidth-Adaptive Dynamic Quantization (BADQ)

---

**Input:** Quantized model parameters  $\{q_i^t\}$ ; scale  $S_t$ ; global model  $w^t$ ; sample sizes  $\{n_i\}$ ; sensitivity  $\lambda$ ; fixed-point precision  $h$ .

**Output:** Weighted integer model parameters  $\{u_i^t\}$ ; integer weights  $\{\bar{a}_i^t\}$ .

```

1: for each client  $C_i \in S_t$  in parallel do.
2:   Dequantize  $q_i^t$  using Eq. (9) and reconstruct the local model using Eq. (13).
3:   Compute the normalized distance  $d_i^t$  by Eq. (14).
4:   Compute the distance coefficient  $\rho_i^t$  by Eq. (15).
5:   Compute the aggregation weight  $a_i^t$  by Eq. (16).
6:   Encode  $a_i^t$  as the integer weight  $\bar{a}_i^t$  by Eq. (17).
7:   Compute the weighted integer update  $u_i^t$  by Eq. (18).
8:   Retain  $(u_i^t, \bar{a}_i^t)$  locally.
9: end for
10: return  $\{u_i^t, \bar{a}_i^t\}$ .

```

---

384 **Algorithm 3** Distance-Aware Integer Weighting (DAIW)

---

385 ● **Chaotic masking encryption for privacy protection**

386 Second, each client generates a random mask and applies it to the processed integer. The first  
387 half of Algorithm 4 generates pairwise chaotic noise from shared keys to resist privacy inference  
388 attacks such as gradient inversion. The client directly adds these random masks to its weighted  
389 integer update. This lightweight encryption requires little computation and hides plaintext  
390 parameters to protect local privacy.

391 ● **Secure summation and aggregation for the updated model**

392 Finally, the server aggregates the masked integers. The second half of Algorithm 4 uses the  
393 opposite signs of pairwise masks to cancel all masks during addition. The server gets the exact

394 aggregate sum without accessing any individual plaintext update. It then dequantizes the sum with  
 395 the shared scale and model parameters the global model. One EPACE-FL framework round is  
 396 completed with both communication efficiency and privacy protection.

---

**Input:** Weighted model parameters  $\{u_i^t\}$ ; integer weights  $\{\tilde{a}_i^t\}$ ; pairwise keys  $\{K_{ij}\}$ ; client set  $S_t$ ; round index  $t$ ; scale  $S_t$ ; modulus  $p$ ; global model  $w^t$ .

**Output:** Updated global model  $w^{t+1}$

- 1: Ensure that  $p$  satisfies the no-wraparound condition in Eq. (28).
  - 2: for each client  $C_i \in S_t$  in parallel do
    - 3: //Lightweight masking with minimal computation workload  $\omega$  by Eq.(29)
    - 4: Derive the domain-separated pairwise seeds by Eq. (19).
    - 5: Generate the update and weight masks by Eq. (20).
    - 6: Construct the signed total masks by Eq. (21).
    - 7: Mask  $u_i^t$  and  $\tilde{a}_i^t$  separately by Eq. (22).
    - 8: //Transmit payload  $\mu$  bounded by communication latency  $T_{comm}$  by Eq.(30)
    - 9: Upload only the masked pair  $(\tilde{u}_i^t, \tilde{a}_i^t)$ .
  - 10: end for
  - 11: if any edge device disconnects due to unstable edge computing then
    - 12: Abort the round or invoke a dropout-recovery mechanism.
  - 13: end if
  - 14: Aggregate the masked uploads by Eq. (23).
  - 15: Decode the aggregate sums by Eqs. (24)-(26) and update global model by Eq.(27).
  - 16: //Evaluate total system latency  $T_{total}$  by Eq. (31)
  - 17: if  $T_{total} > T_{max}$  then record edge computing bottleneck warning.
  - 18: return  $w^{t+1}$ .
- 

397 **Algorithm 4** Chaotic Masking and Secure Aggregation(CMSA)

## 398 5. Experimental Results and Analysis

399 This section evaluates EPACE-FL framework in terms of model utility, communication  
 400 efficiency, dynamic bandwidth adaptation, system execution overhead, privacy protection, and

module contribution. All methods follow the same data split, client sampling strategy, and number of training rounds. Each reported result is the average of three runs with different random seeds.

## 5.1 Experimental Setup

### ● Datasets, Models, and Data Partitioning

This paper employs two public image-classification datasets, MNIST and CIFAR-10. The MNIST dataset is publicly available at <https://github.com/cvdfoundation/mnist>, and the CIFAR-10 dataset is publicly available at <https://www.cs.toronto.edu/~kriz/cifar.html>. MNIST contains 60K training images and 10K test images, whereas CIFAR-10 contains 50K training images and 10K test images. MNIST contains 60K training images and 10K test images. They also pair with LeNet-5. Meanwhile CIFAR-10 contains 50K training images and 10K test images. They also pair with ResNet-18. In the IID setting, the system assigns training samples randomly and uniformly to clients, while the Non-IID setting adopts a Dirichlet distribution with heterogeneity parameter  $\alpha = 0.5$  to generate label proportions. Overall, this setup covers efficient models, deep models, IID data, and statistically heterogeneous data.

### ● Training Parameters and Hardware/Software Environment

The experimental setup contains 20 clients and randomly selects 10 online clients in each round. Each client performs  $E = 5$  local epochs with a batch size of 64. And all methods adopt SGD. The initial learning rate is 0.01 for MNIST and 0.1 for CIFAR-10. And CIFAR-10 adopts cosine annealing. The aggregation server runs on an NVIDIA RTX 4090 workstation, while the client processing time is independently measured on a Raspberry Pi microcomputer. Using the Raspberry Pi realistically simulates the severe hardware capability constraints of typical edge

computing nodes. Network bandwidth varies from 10 Mbps to 5 Mbps, 5 Mbps to 1 Mbps, and then back to 5 Mbps. The base latency is 15 ms. And the candidate dynamic quantization bit widths are {4, 6, 8}.

**Table 1.** Experimental parameter settings

| Parameter                         | Setting                               |
|-----------------------------------|---------------------------------------|
| Dataset / Model                   | MNIST / LeNet-5; CIFAR-10 / ResNet-18 |
| Total clients / Clients per round | 20 / 10                               |
| Data partition                    | IID; Non-IID, Dirichlet alpha = 0.5   |
| Local training                    | E = 5; Batch Size = 64; SGD           |
| Global training rounds            | 200                                   |
| Initial learning rate             | 0.01 (MNIST); 0.1 (CIFAR-10)          |
| Dynamic bit width                 | {4, 6, 8} bit                         |
| Bandwidth / Base latency          | 10→5→1→5 Mbps/15 ms                   |
| Server / Client                   | NVIDIA RTX 4090 / Raspberry Pi 4      |

## ● Compared Algorithms and Evaluation Metrics

The comparison includes FedAvg algorithm which uses plaintext model parameters. The comparison also includes DP-FedAvg algorithm which applies differential privacy noise during training. The Masking+FHE algorithm combines masking with fully homomorphic encryption. The AGQ-FL algorithm applies adaptive gradient quantization. The proposed EPACE-FL framework uses DQCM algorithm for model aggregation.

The evaluation measures final test accuracy. It also measures the communication rounds required to reach the target accuracy. The evaluation measures the one-round client uplink volume.

It also measures cumulative bidirectional communication volume. The evaluation measures one-round client processing time and the one-round server processing time.

The evaluation further considers MSE in the EGIA attack [10]. The evaluation considers PSNR in the same attack. The evaluation considers SSIM at the same attack. The evaluation also considers the defense success rate at the same attack.

## 5.2 Comprehensive Evaluation and Performance

### ● Model Utility and Convergence Performance

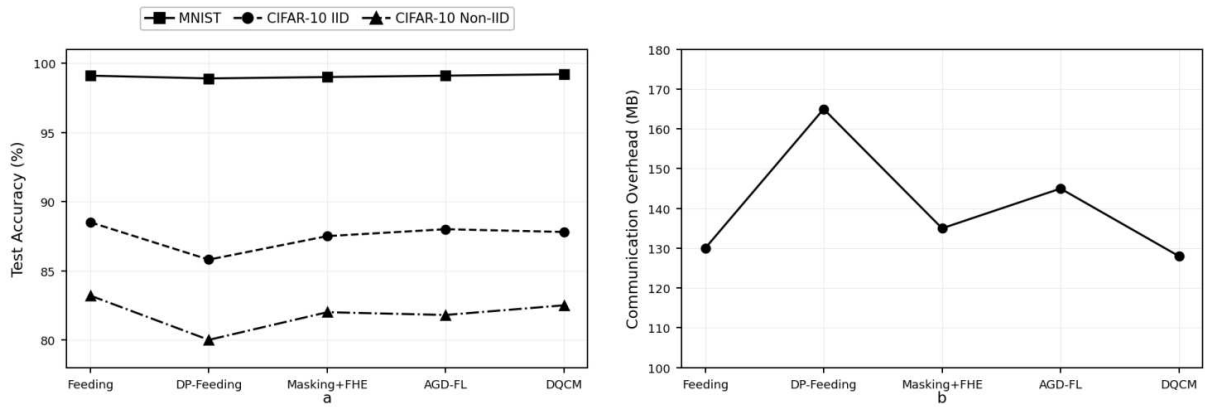


**Fig. 3** Model utility and convergence performance of different methods. **a.** Comparison of final model utility across different algorithms **b.** Comparison of communication rounds required to reach the target accuracy

Figure 3 illustrates the model utility and convergence performance of different methods. In terms of final test accuracy, the baseline FedAvg achieves the maximum accuracy, while DP-FedAvg produces the minimum accuracy. This paper proposes EPACE-FL framework. It also develops DQCM algorithm within this framework. EPACE-FL framework employs DQCM for model aggregation. The convergence rate of EPACE-FL framework is comparable to that of FedAvg algorithm. The final accuracy of EPACE-FL framework is significantly higher than the

final accuracies of DP-FedAvg algorithm and AGQFL algorithm. This performance difference stems from their distinct mechanisms. DP-FedAvg algorithm continuously introduces noise during training. This noise reduces model utility. DQCM algorithm employs a distance-aware aggregation mechanism. This mechanism dynamically adjusts the aggregation weights. The weight adjustment mitigates quantization errors. This effect preserves model precision.

## ● Evaluation of Communication Overhead

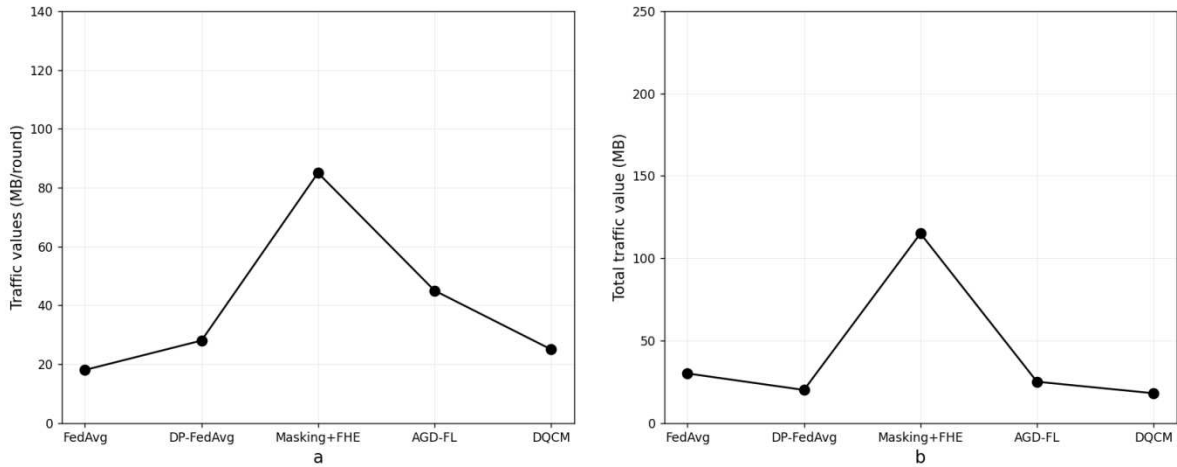


**Fig. 4** Communication overhead of different methods in dynamic bandwidth environments. **a.** Comparison of average one-round client uplink communication overhead **b.** Comparison of cumulative bidirectional communication overhead

Figure 4 presents the communication overhead of different methods in dynamic bandwidth environments. Plaintext FedAvg algorithm generates the maximum cumulative bidirectional traffic of 177.08 GB, whereas the proposed DQCM algorithm achieves the minimum traffic of 119.11 GB. These data indicate a negative correlation between our method and communication costs with the EPACE-FL framework reducing the total communication overhead by 29.3 % compared to FedAvg algorithm. This significant reduction occurs because DQCM algorithm

dynamically maps 32-bit floating-point parameters to low-bit integers, effectively compressing the payload before transmission without suffering from the heavy ciphertext expansion typical of homomorphic encryption.

#### ● Adaptability to Dynamic Edge Bandwidth

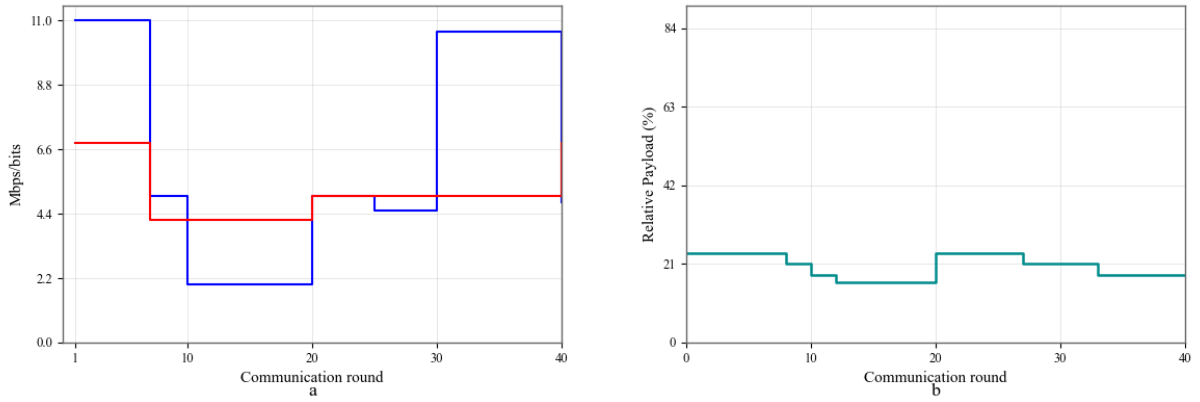


**Fig. 5** Dynamic quantization behavior of EPACE-FL with periodic bandwidth variations. **a.**

Network-aware dynamic bit-width selection. **b.** Payload adaptation after dynamic quantization

AS figure 5 shows, The maximum quantization bit width of 8 bits is selected when the bandwidth is at its peak of 10 MB/s, while the minimum bit width of 4 bits is triggered during severe network congestion of 1 Mb/s. Experimental results demonstrate a positive correlation between the transmission payload and available network bandwidth. The assigned quantization bit width decreases synchronously with the reduction of bandwidth. The EPACE-FL framework achieves this adaptability through active network state monitoring in each communication round. The application of a lower bit width directly prevents transmission timeouts and network congestion during poor network conditions.

#### ● Analysis of System Execution Overhead

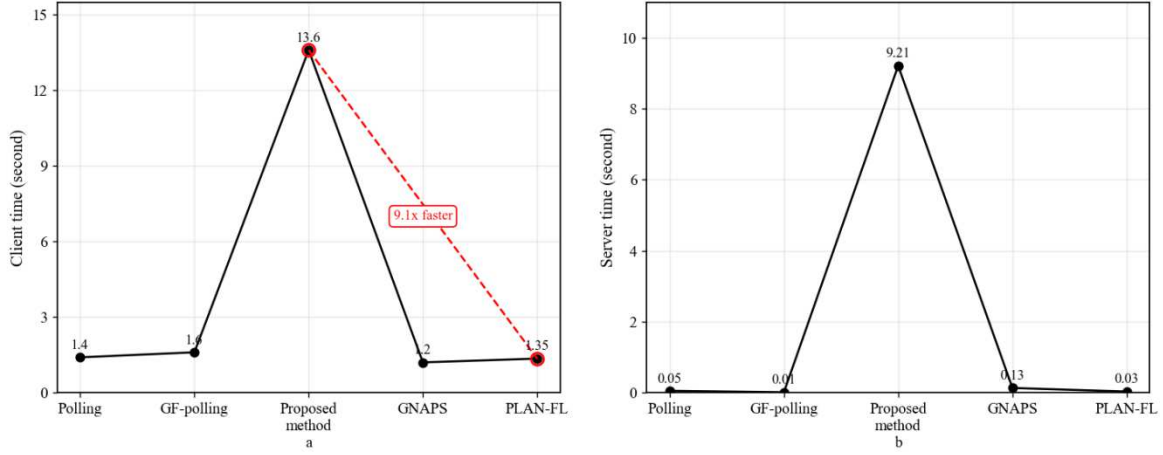


**Fig. 6** Per-round system execution time of different methods. **a** Raspberry Pi 4 client. **b**. RTX 4090 aggregation server

Figure 6 details the per-round system execution time for both the Raspberry Pi 4 client and the RTX 4090 server. Masking+FHE requires the maximum execution time on both devices, whereas the plaintext FedAvg algorithm requires the minimum execution time. The DQCM algorithm incurs a marginal computational overhead over the FedAvg algorithm. The execution speed of DQCM algorithm significantly surpasses Fully Homomorphic Encryption (FHE). The EPACE-FL framework implements efficient additive chaotic masking within the integer domain. This integer-based design successfully eliminates the complex polynomial multiplications and heavy computational burdens of FHE.

#### ● Effectiveness of privacy protection

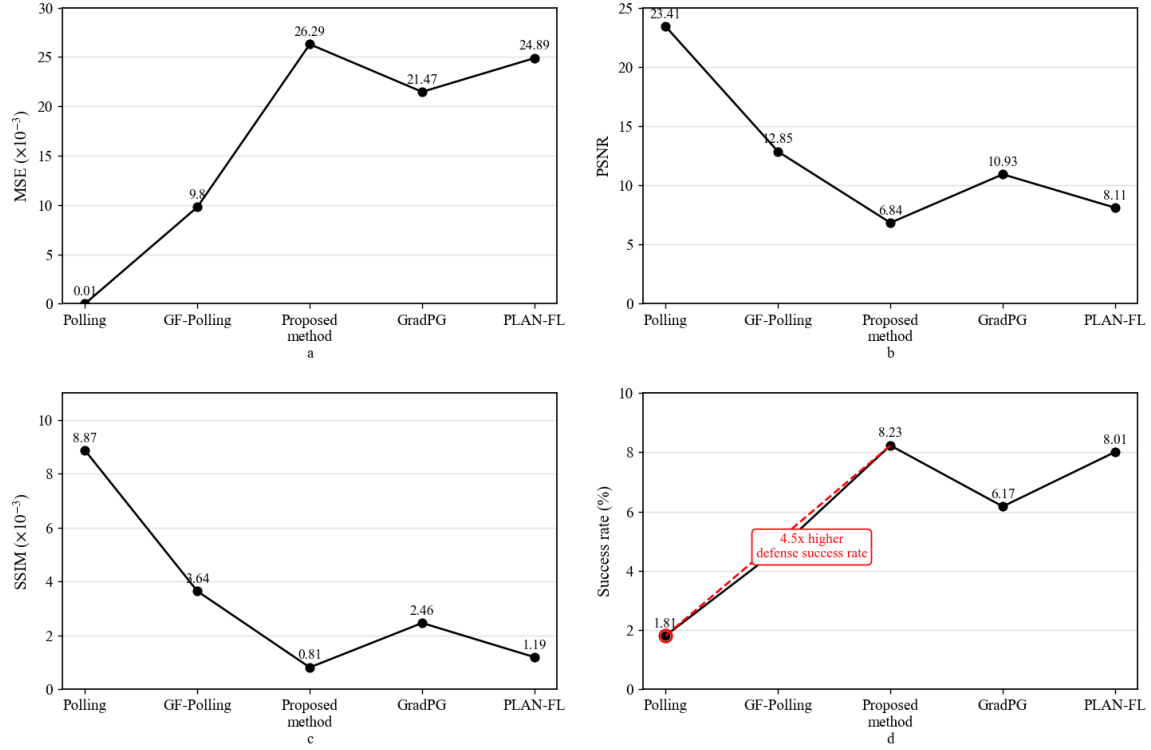


**Fig. 7** Defense effectiveness of different methods against the EGIA gradient inversion attack **a** MSE in the EGIA Attack. **b** PSNR in the EGIA Attack **c** SSIM in the EGIA Attack **d** Defense Success Rate against the EGIA Attack

Figure 7 evaluates the privacy protection effectiveness of different methods against the EGIA model parameters inversion attack. The proposed DQCM algorithm achieves the maximum defense success rate of 89.4 %, whereas the plaintext FedAvg algorithm exhibits the minimum defense success rate of 8%. The data demonstrates that the DQCM algorithm effectively disrupts the attack, outperforming the DP-FedAvg algorithm by 41.6% in protection success rate. The EPACE-FL framework applies pairwise chaotic masks to the uploaded parameters. This masking mechanism completely obfuscates the plaintext gradients at the client side. The robust obfuscation fundamentally prevents backward mathematical deduction by malicious attackers.

### 5.3 Ablation Experiment

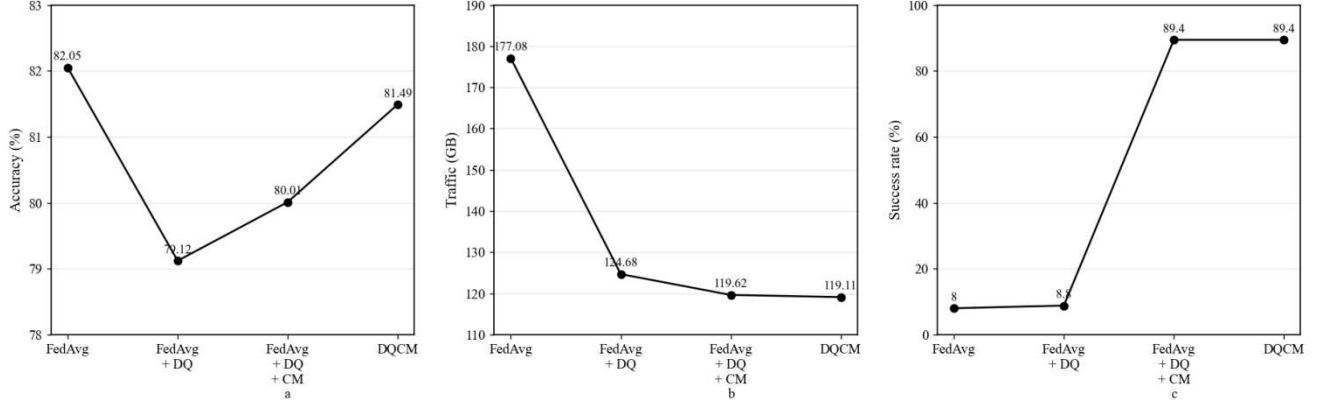


**Fig. 8** Ablation study of the components in the EPACE-FL framework. **a** Utility contribution. **b** Compression contribution c. Privacy contribution

Figure 8 shows the ablation study of the individual components within EPACE-FL framework. For the traffic metric, FedAvg algorithm reaches the maximum value of 177.08 GB, while DQCM algorithm reaches the minimum value of 119.11 GB. For the defense success rate, the DQCM algorithm achieves the maximum value of 89.4 %. The relationship reveals that applying Dynamic Quantization alone reduces traffic but causes accuracy drops, whereas integrating both Dynamic Quantization and Chaotic Masking alongside distance-aware weights restores accuracy while maximizing security. The separation of compression and encryption leads to sub-optimal performance. The proposed DQCM algorithm integrates these two operations. Distance-aware weights effectively mitigate the quantization accuracy loss. The chaotic mask securely encrypts the quantized integers. This cryptographic step perfectly preserves the compressed data size.

## 6. Conclusion and Discussion

Conventional federated learning suffers from insufficient privacy protection and low communication efficiency. To address these limitations, this paper proposes an efficient

privacy-preserving and communication federated learning framework, EPACE-FL. Based on this framework this paper propose the DQCM algorithm. This integrated approach effectively resolves the aforementioned bottlenecks. By synergizing dynamic quantization, distance-aware aggregation, and chaotic masking, EPACE-FL framework achieves highly efficient and secure model aggregation across dynamic and resource-constrained edge computing. Extensive evaluations on MNIST and CIFAR-10 demonstrate that EPACE-FL reduces client uplink and total communication overheads by 58.6% and 29.3%, respectively with a marginal computational delay. Furthermore, it outperforms DP-FedAvg algorithm by 41.6% in defending against privacy inference attacks, while maintaining near-lossless accuracy compared to plaintext training. Despite these compelling results, future work will focus on enhancing the resilience of the framework to dynamic client dropouts and active poisoning attacks, as well as validating its robustness in fully heterogeneous real-world edge environments.

## References

- [1] MCMAHAN B, MOORE E, RAMAGE D, et al. Communication-efficient learning of deep networks from decentralized data. Proceedings of AISTATS. PMLR, 2017, 54: 1273-1282.
- [2] KAIROUZ P, MCMAHAN H B, AVENT B, et al. Advances and open problems in federated learning. Foundations and Trends in Machine Learning, 2021, 14(1-2): 1-210. DOI: 10.1561/22000000083.
- [3] LI T, SAHU A K, TALWALKAR A, et al. Federated learning: Challenges, methods, and future directions. IEEE Signal Processing Magazine, 2020, 37(3): 50-60. DOI: 10.1109/MSP.2020.2975749.
- [4] LIM W Y B, LUONG N C, HOANG D T, et al. Federated learning in mobile edge computing:

547 A comprehensive survey. IEEE Communications Surveys & Tutorials, 2020, 22(3):  
548 2031-2063. DOI: 10.1109/COMST.2020.2986024.

549 [5] ALISTARH D, GRUBIC D, LI J, et al. QSGD: Communication-efficient SGD via gradient  
550 quantization and encoding. Advances in Neural Information Processing Systems. 2017, 30:  
551 1709-1720.

552 [6] HINTON G, VINYALS O, DEAN J. Distilling the knowledge in a neural network. NIPS Deep  
553 Learning Workshop. 2015: 1-9.

554 [7] HAN S, MAO H, DALLY W J. Deep compression: Compressing deep neural networks with  
555 pruning, trained quantization and Huffman coding. International Conference on Learning  
556 Representations. 2016.

557 [8] MICIKEVICIUS P, NARANG S, ALBEN J, et al. Mixed precision training. International  
558 Conference on Learning Representations. 2018.

559 [9] KARIMIREDDY S P, REBJOCK Q, STICH S U, et al. Error feedback fixes SignSGD and  
560 other gradient compression schemes. Proceedings of ICML. PMLR, 2019, 97: 3251-3260.

561 [10] LIANG H, LI Y, ZHANG C, et al. EGIA: An external gradient inversion attack in federated  
562 learning. IEEE Transactions on Information Forensics and Security, 2023, 18: 4984-4995.  
563 DOI: 10.1109/TIFS.2023.3302161.

564 [11] DIMITROV D, BAADER M, MÜLLER M, et al. SPEAR: Exact gradient inversion of batches  
565 in federated learning. Advances in Neural Information Processing Systems. 2024, 37:  
566 106768-106799. DOI: 10.52202/079017-3390.

567 [12] WEI K, LI J, DING M, et al. Federated learning with differential privacy: Algorithms and  
568 performance analysis. IEEE Transactions on Information Forensics and Security, 2020, 15:  
569 3454-3469. DOI: 10.1109/TIFS.2020.2988575.

- [13]CHEON J H, KIM A, KIM M, et al. Homomorphic encryption for arithmetic of approximate numbers. *Advances in Cryptology—ASIACRYPT*. Springer, 2017: 409-437. DOI: 10.1007/978-3-319-70694-8\_15.
- [14]AONO Y, HAYASHI T, WANG L, et al. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 2018, 13(5): 1333-1345. DOI: 10.1109/TIFS.2017.2787987.
- [15]MOHASSEL P, ZHANG Y. SecureML: A system for scalable privacy-preserving machine learning. *IEEE Symposium on Security and Privacy*. 2017: 19-38. DOI: 10.1109/SP.2017.12.
- [16]WAGH S, GUPTA D, CHANDRAN N. SecureNN: 3-party secure computation for neural network training. *Proceedings on Privacy Enhancing Technologies*, 2019, 2019(3): 26-49. DOI: 10.2478/popets-2019-0035.
- [17]TRUEX S, BARACALDO N, ANWAR A, et al. A hybrid approach to privacy-preserving federated learning. *Proceedings of the ACM Workshop on Artificial Intelligence and Security*. 2019: 1-11. DOI: 10.1145/3338501.3357370.
- [18]Sav S, Pyrgelis A, Troncoso-Pastoriza JR et al (2021) POSEIDON: Privacy-preserving federated neural network learning. *Network and Distributed System Security Symposium (NDSS 2021)*. <https://doi.org/10.14722/ndss.2021.24007>.
- [19]BONAWITZ K, IVANOV V, KREUTER B, et al. Practical secure aggregation for privacy-preserving machine learning. *Proceedings of ACM CCS*. 2017: 1175-1191. DOI: 10.1145/3133956.3133982.
- [20]CHILLOTTI I, GAMA N, GEORGIEVA M, et al. TFHE: Fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 2020, 33: 34-91. DOI: 10.1007/s00145-019-09319-x.
- [21]CUI L, WU X. ALDP-FL for adaptive local differential privacy in federated learning.

Scientific Reports, 2025, 15: 26679. DOI: 10.1038/s41598-025-12575-6.

[22]ZHU L, SONG H, CHEN X. Dynamic privacy budget allocation for enhanced differential privacy in federated learning. Cluster Computing, 2025, 28(15): 999. DOI: 10.1007/s10586-025-05699-x.

[23]WANG D, GAO Y, PANG S, et al. FedMPS: A robust differential privacy federated learning based on local model partition and sparsification for heterogeneous IIoT data. IEEE Internet of Things Journal, 2025, 12(10): 13757-13768. DOI: 10.1109/JIOT.2025.3536035.

[24]LIU H, HE F, CAO G. Communication-efficient federated learning for heterogeneous edge devices based on adaptive gradient quantization. IEEE INFOCOM. 2023: 1-10. DOI: 10.1109/INFOCOM53939.2023.10228970.

[25]QIN X, YANG X, TANG X. Efficient Byzantine-robust privacy-preserving federated learning via dimension compression. IEEE Transactions on Information Forensics and Security, 2026, 21: 2596-2609. DOI: 10.1109/TIFS.2026.3671104.

[26]NGUYEN M H, NGUYEN C H, NGUYEN D N, et al. Adaptive quantization and differential privacy federated learning framework. IEEE Transactions on Mobile Computing, 2026, 25(7): 10335-10350. DOI: 10.1109/TMC.2026.3665886.

[27]PARK S, LEE J, HARADA K, et al. Masking and homomorphic encryption-combined secure aggregation for privacy-preserving federated learning. Electronics, 2025, 14(1): 177. DOI: 10.3390/electronics14010177.

[28]LI G, ZHANG Z, DU R. LVSA: Lightweight and verifiable secure aggregation for federated learning. Neurocomputing, 2025, 648: 130712. DOI: 10.1016/j.neucom.2025.130712.

[29]LI J, ZHOU Z, TIAN Y, et al. FedCPA: A two-layer secure aggregation federated learning with lightweight privacy protection for Industry 4.0. Cybersecurity, 2026, 9(1): 152. DOI:

616 10.1186/s42400-026-00575-6.

## 617 **Data availability**

618 The datasets analyzed in this study are publicly available. The MNIST dataset is available at  
619 <https://github.com/cvdfoundation/mnist>, and the CIFAR-10 dataset is available at  
620 <https://www.cs.toronto.edu/~kriz/cifar.html>. No new raw datasets were generated in this study.  
621 The client-level data partitions and experimental settings are described in the manuscript and are  
622 available from the corresponding author upon reasonable request.

623 **Funding** This study was funded by Scientific Research Startup Fund Project for High-level  
624 Talents of Xi'an International University(XAIU202411)

625 **Author contributions** : Chao-Hsien Hsieh: Conceptualization, Formal analysis, Writing –  
626 Review & Editing. Peidong Wang: Methodology , Software, Validation, Data  
627 Curation, Writing – Original Draft, Visualization.