Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Quantization-Based Chained Privacy-Preserving Federated Learning

Ya Liu

liuya@usst.edu.cn

University of Shanghai for Science and Technology

Shumin Wu

University of Shanghai for Science and Technology

Yibo Li

University of Shanghai for Science and Technology

Fengyu Zhao

Shanghai Publishing and Printing College

Yanli Ren

Shanghai University

Article

Keywords:

Posted Date: April 15th, 2025

DOI: https://doi.org/10.21203/rs.3.rs-5858510/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Version of Record: A version of this preprint was published at Scientific Reports on May 16th, 2025. See the published version at https://doi.org/10.1038/s41598-025-01420-5.

Quantization-Based Chained Privacy-Preserving Federated Learning

Ya Liu[1,2,*], Shumin Wu[1], Yibo Li[1], Fengyu Zhao[3], and Yanli Ren[4]

[1]The Department of Computer Science and Engineering, University of Shanghai for Science and Technology, Shanghai, 200093, China
[2]Lion Rock Labs of Cyberspace Security, HKCT Institute of Higher Education , Hong Kong, China
[3]The Department of Information and Intelligence Engineering, Shanghai Publishing and Printing College, Shanghai, 200093, China
[4]The School of Communication and Information Engineering, Shanghai University, Shanghai, 200444, China
[*]liuya@usst.edu.cn

ABSTRACT

Federated Learning (FL) is an advanced distributed machine learning framework crucial in protecting data privacy and security. By enabling multiple participants to train models while keeping their data local collaboratively, FL effectively mitigates the risks associated with centralized storage and sharing of raw data. However, traditional FL schemes face significant challenges regarding communication efficiency, computational costs, and privacy preservation. For instance, its communication and computational overhead in edge computing scenarios is often excessively high, hindering real-time applications. This paper proposes an innovative federated learning framework, Q-Chain FL, integrating quantization compression techniques into a chained FL architecture. This Q-Chain FL scheme adopts efficient compression and transmission of model parameter differences at the user node and executes seamless decompression and aggregation at the server node. Experiments on several publicly available datasets, including MNIST, CIFAR-10, and CelebA, demonstrate low communication and computational overhead, fast convergence speed, and high security of Q-Chain FL. Compared to traditional FedAvg and Chain-PPFL, Q-Chain FL reduces communication overhead by approximately 62.5% and 44.7%, respectively. These results underscore the robustness and adaptability of Q-Chain FL in various datasets and real-world learning scenarios.

1 Introduction

In the current era of explosive data growth, machine learning plays a vital role in data mining and is widely applied in numerous fields[1,2]. However, the traditional centralized machine learning paradigm has significant drawbacks. It requires storing many raw datasets centrally on a single central server, which undoubtedly brings serious risks to user data privacy leakage[3,4]. Especially in sensitive areas such as medical diagnosis, the data may contain extremely sensitive personal information that is strictly prohibited from being directly shared with third parties[5]. Moreover, large-scale data transmission imposes a massive burden on network resources, making the application of traditional centralized machine learning in practical scenarios face numerous difficulties.

Federated learning is a promising decentralized learning paradigm to solve these problems and is gradually attracting widespread attention[6,7]. In the federated learning framework, multiple geographically dispersed clients can train machine learning models locally using their raw data without transmitting them to a central server, thus effectively decreasing the risk of data privacy leakage. Specifically, the aggregation server distributes the parameters of the global model to selected clients in each training round. These clients then train local models based on their data and upload the models' parameters to the aggregation server. Subsequently, the server fuses these models' parameters to compute a globally updated model[8]. Through this approach, federated learning achieves model optimization and knowledge fusion while keeping the data local, gradually improving the global model's performance. Among numerous federated learning algorithms, Fedavg[9] is a typical and widely used scheme.

Although federated learning has many advantages, it still faces a series of serious challenges[10]. For example, in edge computing environments[11], limited bandwidth and computational resources greatly restrict the performance of federated learning. Specifically, on the one hand, the frequent communication between selected clients and the aggregation server leads to high communication costs, which becomes a key factor restricting system performance in bandwidth-constrained edge computing scenarios[12–16]. On the other hand, model updating and parameter transfer involve a large quantity of data transfer and computation, which not only increases the computational complexity but also may reduce the convergence of model training. In addition, security issues are also an important aspect that cannot be ignored in federated learning[17]. Li et al. proposed

the Chain-PPFL scheme[18], which adopted a chain-based structure to directly transmit the model parameter one after another and the last user transferred the model parameter to the server. This scheme improves privacy-preserving and communication overhead to some extent. However, the Chain-PPFL also faces high computation costs and security vulnerabilities caused by direct model parameter exchange. Therefore, it is an urgent problem how to balance data security and communication and computation overhead of FL in edge computing environments further[19].

In this paper, we propose the Q-Chain FL scheme, which innovatively combines chain structures with quantization compression techniques to solve problems regarding communication and computational costs and privacy preservation. Specifically, participants in the Q-Chain FL scheme transmit model parameter packages that have undergone quantization difference compression and masking operations rather than the original model parameters. These parameter packages are transferred sequentially among participants and finally transmitted to the central server by the last user on the user list. After receiving the parameter packages, the server performs dequantization and aggregation operations and broadcasts a high-precision global model. This organic combination of quantization operation and chain structure brings numerous advantages to federated learning. In edge computing environments, it not only significantly reduces communication overhead and storage costs but also greatly enhances user privacy and security. Meanwhile, while maintaining model performance, the Q-Chain FL scheme significantly accelerates the convergence speed of the global model, effectively reduces computational complexity, and remarkably improves communication efficiency, achieving a perfect balance between computational efficiency and data security.

Contributions. The contributions can be outlined as follows:

- In edge-computing scenarios, Q-Chain FL integrates quantization and chain structures. It reduces the communication volume by approximately 62.5% and 44.7% compared with FedAvg and Chain-PPFL, respectively.
- Q-Chain FL adopts transmission masking and compressed parameter differences to enhance privacy preservation. In terms of security, it outperforms traditional FL and Chain-PPFL.
- Experiments on the dataset demonstrate that the convergence speed of Q-Chain FL is 33.3% faster than that of traditional FedAvg and 28.6% faster than that of Chain-PPFL.

To intuitively demonstrate the advantages of the Q-Chain FL scheme, we summarize comparisons of Q-Chain FL, FedAvg, and Chain-PPFL in terms of key indicators such as communication volume, security, and convergence speed in Table 1. Clearly, Q-Chain FL achieves low communication and computation costs and high security while maintaining or enhancing the accuracy of FL.

Table 1. Comparison of FL schemes on key metrics

Scheme	Communication Volume (GB/round)	Security[1]	Convergence Speed (epochs)[2]
FedAvg[9]	High (5.6)	Low	30
Chain-PPFL[18]	Medium (3.8)	Moderate	28
Q-Chain FL (Proposed)	Low (2.1)	High	20

[1] Security-High indicates compressed model parameter differences transmission instead of direct model parameter transmission.
[2] Convergence Speed is the number of rounds for the model accuracy of 90% on the MNIST dataset.

Organization. This paper is structured as follows. Section II gives related research. Section III provides an in-depth exploration of the architecture and implementation of Q-Chain FL. Section IV compares the Q-Chain FL scheme to previous ones. Section V conducts detailed comparative experiments and interprets the results. Finally, section VI presents a comprehensive summary of this paper.

2 Related Work

This section highlights classic FedAvg architecture and emphasizes some challenges of applying chain-structured federated learning in edge computing environments. It also explores quantization techniques to solve communication and storage overheads in large-scale federated learning.

2.1 Federated Learning

FL, a vital implementation of distributed machine learning, represents a research domain of paramount interest. It aims to facilitate global model training involving multiple participants while safeguarding the privacy of sensitive data. The common

architecture of FL, FedAvg, has sparked widespread research fervor. It establishes a collaborative framework with a centralized server and multiple participants and adopts an iterative model aggregation strategy. Within the FedAvg framework, the process commences with the centralized server's random initialization of a global model. Subsequently, the global model chooses a subset of participants for model training by random or weighted selection techniques.

Following participant selection[20,21], the server transfers the existing global model to the chosen clients. These participants leverage their own data to train the global model locally, progressively refining the local model's parameters. Subsequently, the participants transmit the model parameters acquired from their training to the server.

Upon receiving these parameters, the central server aggregates them by adopting techniques such as simple averaging, weighted averaging[22,23], etc. This amalgamation process is pivotal in generating a novel global model. FedAvg diligently pursues global model optimization by orchestrating a sequence of actions that includes multi-modal distribution, local training, and model aggregation. This process will not terminate until predefined criteria are met, such as a specified number of iteration rounds or halting conditions.

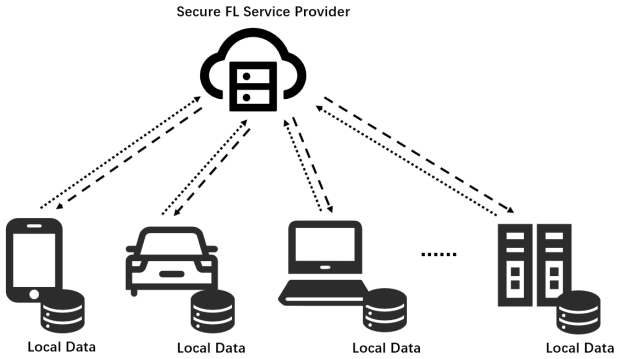


Figure 1. Federated learning training models for devices in edge computing scenarios.

However, cutting-edge research such as Chain PPFL proposes a level-by-level method of transferring model parameters, where participants locally train after receiving the parameters of the previous level and transmit the updated parameters to the next level. The technique successfully combines chain-structured FL with edge-computing scenarios. In the edge-computing system, edge devices (e.g., Teslas, PCs, mobile devices, etc.) act as participants and train local data for a global model. The central server aggregates local training model parameters to form comprehensive global models, enabling knowledge sharing and model evolution (see Figure 1). However, FL demands each edge device's high computational and storage capacity and needs to maintain high communication efficiency between participants. Therefore, it is the core task of FL to solve these complex problems.

2.2 Quantization Technology

In recent years, researchers have shown a strong interest in optimizing the communication efficiency of distributed machine learning, especially in quantization-based methods. For example, Zhou et al. proposed a quantization technique for deep convolutional networks, which achieved efficient network inference through pruning, training quantization, and Huffman coding[24]. However, the pruning operation may cause the loss of some important information. Courbariaux et al. explored a binary connectivity scheme that uses binary weights for training, significantly reducing the network size and maintaining the performance of the model to a certain extent[25]. Nevertheless, the overly simple binary representation may not be able to express the complex information of the model accurately. Ullrich et al. proposed a soft weight-sharing approach to reduce the network size by sharing weights within the network and optimize storage and computational efficiency[26]. However, weight sharing may result in inconsistent model updates across different nodes. Hubara et al. investigated a quantized neural network approach using low-precision weights and activation values, achieving model compression by reducing the bit-width of the weights and activation values[27]. However, current quantization techniques face issues such as computational complexity during the dequantization process, information loss, degradation of model accuracy, and compatibility problems in the fusion of models from different nodes. These issues limit their effectiveness and performance improvement in the model aggregation of federated learning.

3 System Architecture and Method

This section describes the system's design goals and presents the Q-Chain FL scheme. This scheme skillfully integrates chain-structured federated learning with quantization techniques to address computational and storage challenges in resource-

Table 2. Symbol notations

Notation	Definition
k	the k-th user
t	global model training rounds
U	count the number of participants
C	set of users selected for training
P_0^t	pseudo-random generator
$\mathbf{w}_k^r$	the model update of user k in round r
Q_t	t-bit quantizer in the Q-Chian FL
Q_t^{-1}	t-bit dequantizer in the Q-Chian FL
$\mathbf{r}_i^t$	vector of blinding factors of user
$time$	countdown timer for controlling the duration of round
$Package_k^t$	quantitative difference modeling package

constrained environments. Subsequently, we explore the application of quantization techniques in federated learning, especially in optimizing model aggregation and communication efficiency. Finally, we detail the execution steps of the scheme, including the initialization process, the interaction process between user nodes and the server node, and the method for obtaining the final global model. Table 2 describes the key model's parameters and notations.

3.1 Design Goals

The traditional chain-structured federated learning model has multiple problems. First, this model limits the speed of the model's parameter transfer, which negatively impacts overall performance. Second, an imbalance in training may make the model update for sure participants slow or insufficient. In addition, there is a potential risk of privacy breaches during the process of information transmission, which may lead to disclosing sensitive information. On the other hand, it does not continue until previous participants complete model updates, which wastes computing resources. In summary, traditional chain-structured federated learning has some problems regarding communication efficiency, storage burden, privacy preservation, and computational burden.

This paper designs an optimal quantization compression technique compatible with chained structures to achieve improvements in the following:

- **Improving the communication efficiency.** In traditional chain-structured federated learning, the cascading transmission of model parameters may lead to the gradual accumulation of communication overhead, affecting communication efficiency. However, the Q-Chain FL will adopt differential quantization compression technology to process the model parameters during local transmission, thereby reducing communication costs and ensuring smooth communication.
- **Reducing the storage and computation burden.** During each training round, participants must store the transmitted model parameters, which poses a challenge for edge devices with limited resources. The Q-Chain FL scheme will devise a quantitative compression strategy that effectively reduces the computational complexity of model parameter transmission and aggregation, significantly reducing the storage requirements for model parameters. These improvements encourage a wider range of devices to participate in federated learning.
- **Enhanced privacy preservation.** In federated learning, it is possible to leak data privacy during the direct transmission of model parameters. In this paper, we will add masks to local model parameters and apply quantization compression techniques to confuse these parameters. In addition, users will only transmit model quantization differential compressed packages, thereby improving users' data security. This method will solve the high-cost problem caused by existing privacy preservation technologies and reduce the possibility of privacy leakage while maintaining the model's accuracy.

3.2 System Architecture

This section discusses in detail the framework of Q-Chain FL, shown in Figure 2. The Q-Chain FL model integrates chain-structured federated learning and quantization techniques to address various challenges in resource-constrained environments. This system contains the server and users. The server mainly determines the list of users participating in training and broadcasts it along with initial global weights. It also generates a random mask to initialize the package and sends it to the first user in the user list. After receiving the package the last user sends, the server executes parameter aggregation and returns the global model parameter to the participants. The participants conduct local model training, quantify and compress local model parameters, receive packages sent by other users, merge their compressed packages, and send them to the following user. The last user must send the compressed package back to the server. The Q-Chain FL scheme consists of the following steps:

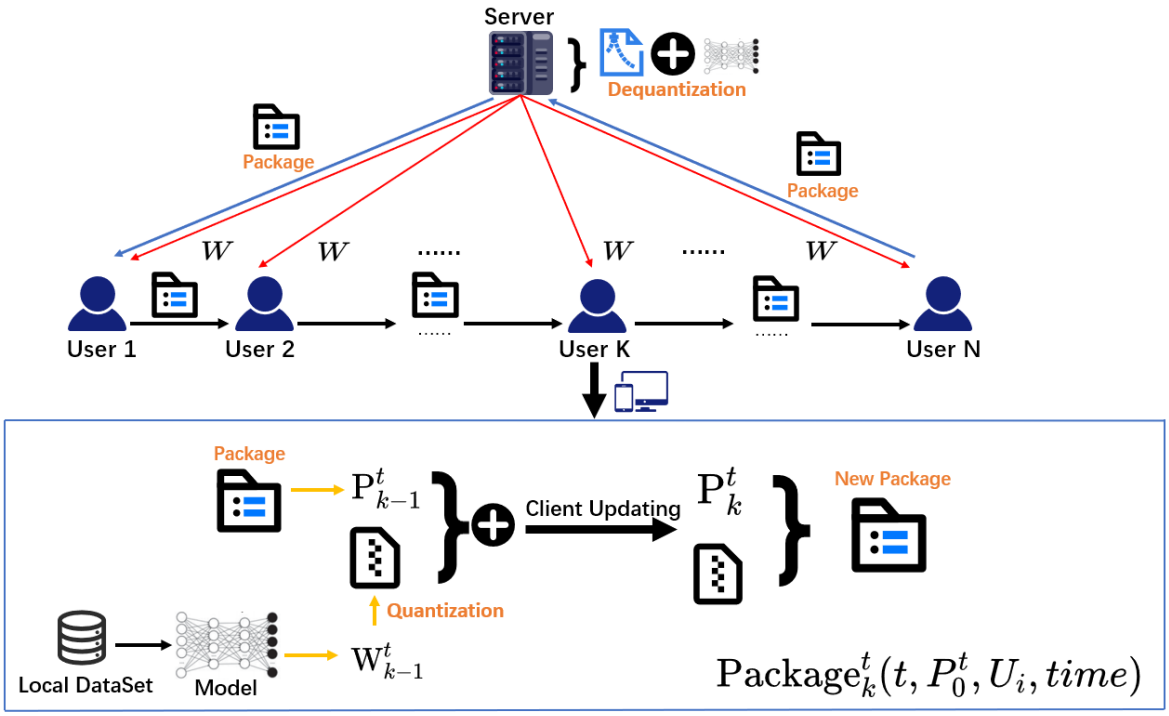


Figure 2. The framework of Q-Chain Federated Learning.

- Step 1: (Initialization) The server initializes the global model and incorporates masks.
- Step 2: (Participant Selection) The central server adopts a weighted or randomized method to select a subset of participants and determine the list of individuals engaged in model training.
- Step 3: (Model Distribution) The server transfers the current global model to the chosen participants.
- Step 4: (Local Model Training) Participants train the received global model locally using local data. Each participant trains on the local dataset for several iterations to update their own model parameters. Then, these parameters are quantitatively compressed and transferred to the next participant until they reach the last participant in the user list.
- Step 5: (Model Aggregation) The last participant in the user list uploads the final package of model parameters processed by quantization compression to the server. Subsequently, the server generates a new global model by aggregating the local parameters achieved by unmasking and quantizing operations.
- Step 6: (Iteration) Iterate steps 3 to 5 to execute numerous cycles of local training, model aggregation, and model distribution until the model iterates for the specified number of rounds or satisfies other termination criteria.

This process realizes global model training in federated learning while protecting user data privacy. Through local training and parameter aggregation, model updates are propagated to the entire system, achieving the goal of federated learning. Meanwhile, we can generate the user list to improve the system's performance by adopting user-selection strategies, such as distance-based and reputation-based methods etc.

The Q-Chain FL scheme randomly selects the next node of the user list for communication in each round. We will describe the user-to-user and user-to-server communication in Section 3.4. During each training session, the server generates an initial package and sends it to the first participant on the user list. This package contains mask information that the user does not know. Then, the user compresses the model locally, merges it into a new package, and sends it to the following user. To enhance user privacy preservation, we adopt a strategy of combining a single mask with quantization compression, which not only effectively resists the potential risks brought by user collusion but also ensures that users cannot obtain any valid information except for the original message during the package transmission process. With these privacy preservation measures, our scheme can ensure communication efficiency while protecting users' privacy to the utmost extent.

3.3 Federated Learning With Quantization

This section proposes a novel quantization technique compatible with our system's parameter compression and model aggregation. Initially customized for homomorphic encryption, this technique quantifies values into r bit-signed integers and introduces zero-symmetric quantization ranges and fractional value ranges to ensure that values of opposite signs cancel each other out in the aggregation process. We will describe the technique in the following.

Perform the following operations on the r-bit quantizer Q_r. For any scalar $a \in \mathbb{D}$, let $abs(a)$ denote the absolute value of a and $round(a)$ denote the standard rounding of a. $sgn(a) \in \{-1, 1\}$ represents the sign of a, where $sgn(0) = 1$. The quantizer Q_r quantizes x in the range $[-D, D]$ to an integer in the range $[-(2^{r-1}-1), 2^{r-1}-1]$, shown in Equation (1).

$$Q_r(x) = sgn(x) \cdot round(abs(x) \cdot \frac{2^{r-1}-1}{D}) \ , \tag{1}$$

where D is a constant that determines the range of the quantized value. Given the quantized value, we can also compute the quantized value $Q_r^{-1}(x)$ by the following formula (2).

$$Q_r^{-1}(x) = sgn(Q_r^{(x)}) \cdot \frac{abs(x) \cdot D}{2^{r-1}-1} \ . \tag{2}$$

This formula can invert the quantization operation and provide an approximation of the original value x.

The Q-Chain FL scheme uses this quantization technique to improve communication efficiency by directly compressing the quantized values. Since model differences are easier to compress, the quantization operation is performed on the difference Δ_k^t between the local training model $\mathbf{w}_k^t$ and the global model $\mathbf{w}^{t-1}$ in the previous round, called the model parameter difference, instead of on parameters of the local training model. To ensure that the input to the quantizer is in the range $[D/2, D]$, the Q-Chain FL scheme trims the values in Δ_k^t. The threshold D can be preset according to commonly used datasets. Let values greater than D and values less than $-D$ equal D and $-D$, respectively.

It is essential to prevent the overflow of quantized values during the aggregation process because the central server aggregates quantized values from multiple users. To prevent overflow, we can narrow the input range of quantization to $[-n \cdot D, n \cdot D]$, where n is the number of clients involved in the aggregation. By narrowing the input range, each quantized value is decreased by a factor of n, thus avoiding the overflow problem.

In addition, we need to determine which values should be negative during dequantization. The cloud server must obtain this information before performing the dequantization. If the server checks that the value of a in Δ_k^t is greater than $2^{r-1}-1$, its original value a' equals $a - 2r$. Otherwise, the value should be positive or zero. The server can perform the dequantization process by checking and converting the values that are considered negative.

3.4 Q-Chain FL Algorithm

The Q-Chain FL is based on general assumptions that all participants are honest and curious, and the user-to-user and user-to-server transmission channels are secure and reliable. The algorithm 1 describes in detail the execution of Q-Chain FL in each round.

1) Initialization. The pseudo-random generator (PRG) maps the seeds to a pseudo-random sequence. Based on it, the cloud server generates a random number $\mathbf{PRN}_r$ as a mask and sends an initial global model $\mathbf{w}_0$ to each client, where $\mathbf{PRN}_r \in \mathbb{R}^d$ and d are the dimensions of $\mathbf{w}$. We expand the input into an n-dimensional vector, denoted δ_r by $\mathbf{PRN}_r$. Let P_0^t equal $\mathbf{PRN}_r$. Then, the server creates a package $\mathbf{Package}_0^t(t, P_0^t, U, time)$, where U represents the number of users and $time$ is a countdown timer to control the duration of each round. Based on the above assumptions, the node does not need to inspect the software package. The cloud server transfers the initial package to the first randomly selected client (called User 1) currently participating in training.

2) User Nodes. The first randomly selected user in the t round is called User 1 and has locally optimal parameters $\mathbf{w}_1^t$. The training process for the local model parameters is similar to that of FedAVG. Each user k uses the SGD (stochastic gradient descent) algorithm to train the local model on their dataset. The initial parameters are $\mathbf{w}_0$. In each epoch i, user k samples a small batch of data from the dataset β, computes the gradient of the loss function $\nabla \ell(\mathbf{w}; \beta)$ on dataset β, and update the user's last - round local model parameter $\mathbf{w}_{i-1}$ to $\mathbf{w}_i$ according to the equation (3).

$$\mathbf{w}_i \leftarrow \mathbf{w}_{i-1} - \eta \nabla \ell(\mathbf{w}_{i-1}; \beta), \tag{3}$$

where η and ℓ are the learning rate and the loss function with respect to the model parameters $\mathbf{w}$, respectively. After repeating this process for E epochs, user k obtains the local model parameters, which will be uploaded.

Algorithm 1 The Detailed Processes of Q-Chain FL

Input: initial aggregate model $\mathbf{w}_0$, maximum number of epochs E, select batches of size β, the learning rate η, the number of global iterations t, loss function $\nabla\ell(\mathbf{w};\beta)$.
Output: the latest model $\mathbf{W}$.

Server Node:
Initialize and broadcast $\mathbf{w}_0$
for each epoch $t = 1,2,3,\ldots$ **do**
  User $\leftarrow$ (randomly select from the set of users)
  send $\mathbf{Package}_0^t(t,P_0^t,U,time)$ to User 1
  for each user k in C **do**
    $P_k^t \leftarrow \mathbf{UserUpdate}(\mathbf{Package}_{k-1}^t)$
  end for
  $\Delta^t \leftarrow (\sum_{k\in C} P_k^t - P_0^t) \bmod 2^r$
  $\mathbf{w}^t = \mathbf{w}^{t-1} + \frac{1}{|C|}\cdot Q_r^{-1}(\Delta^t)$
  Wait until the $\mathbf{Package}_k^t$ is received from the final User N.
end for
User Node:
UserUpdate ($\mathbf{Package}_{k-1}^t$)**:**
Generate the vector $\mathbf{r}_i^t$ of blinding factors
for each local epoch i in E **do**
  for each batch β **do**
    $\mathbf{w}_i \leftarrow \mathbf{w}_{i-1} - \eta\nabla\ell(\mathbf{w}_{i-1};\beta)$
  end for
end for
$\Delta_i^t = \mathbf{w}_k^t - \mathbf{w}^{t-1}$
$\mathbf{w'}_i^t \leftarrow Q_r(\Delta_i^t) + \mathbf{r}_i^t \bmod 2^r$
$P_i^t \leftarrow \mathbf{w'}_\mathbf{i}^t + P_{i-1}^t$
$U \leftarrow U + 1$
update $\mathbf{Package}_k^t(t,P_i^t,U,time)$
if $time$== 0 **then**
  send $\mathbf{Package}_n^t(t,P_n^t,U,time)$ to Server Node
else
  send $\mathbf{Package}_k^t(t,P_i^t,U,time)$ to Next User Node
end if

User k computes the local update $\mathbf{w}_k^t$, and the difference between the local training model parameters and the global model parameters in the previous round, shown in equation (4).

$$\Delta_i^t = \mathbf{w}_k^t - \mathbf{w}^{t-1}, \tag{4}$$

Next, generate the blinding factor vector $\mathbf{r}_i^t$, and calculate

$$\mathbf{w'}_i^t = Q_r(\Delta_i^t) + \mathbf{r}_i^t \ (\mathrm{mod}\ 2^r), \tag{5}$$

followed by updating the parameters P_i^t and the counter U. By updating the package $\mathbf{Package}_i^t(t,P_i^t,U,time)$, the result P_i^t is transmitted to the next user (called User $k+1$), which is randomly selected from the current round of participating users.

If $time \le 0$, user n is the last node and user n should transmit the result P_n^t to the server by updating the package $\mathbf{Package}_n^t(t,P_n^t,U,time)$.

3) Server Node. The server node receives the package $\mathbf{Package}_k^t$ from the last user on the list and removes the initial mask according to the equation (6).

$$\Delta^t \leftarrow (\sum_{k\in|C|} P_k^t - P_0^t) \bmod 2^r. \tag{6}$$

Solve the quantization to calculate the global weights for the next round. Meanwhile, the server updates the global model

parameters according to the equation (7).

$$\mathbf{w}^t = \mathbf{w}^{t-1} + \frac{1}{|C|} \cdot Q_r^{-1}(\Delta^t). \tag{7}$$

Until the end of the server-side iteration rounds and the completion of this federated learning task, the server node gets the final aggregated model $\mathbf{W}$.

4 Analysis and comparison

This section compares the Q-Chain FL with two federated learning schemes FedAVG and Chain-PPFL, and analyzes them from privacy preservation and the communication latency of the model structure.

4.1 Analysis of Privacy Preservation

FedAvg has flaws in potential privacy preservation because it does not use any security measures. It only uploads the updated gradient of the model after localized training instead of the complete model parameters, which causes the risk of information leakage. Attackers may use the gradient information to infer sensitive features of personal data, thus compromising user privacy. In the Chain-PPFL scheme, data exchange between neighboring nodes involves only the addition of random numbers. However, this random number-preserving federated learning architecture faces various challenges regarding privacy, such as the trustworthiness of the random number generation process, the reuse and persistence of random numbers, random number leakage and inference, as well as the traceability of random numbers.

In the Q-Chain FL scheme, we provide a detailed analysis of privacy preservation. This scheme safeguards data privacy through data exchange between neighboring nodes and the utilization of compressed package, represented as $\mathbf{Package}_i^t(t, P_i^t, U, time)$, where $P_i^t = \sum \mathbf{w}_i'^t + \mathbf{PRN}_r$. The random constant $\mathbf{PRN}_r$ is provided by the server node, and $\sum \mathbf{w}_i'^t$ represents the quantized parameter differences of the first i users in the t-th round of the user list. The quantized parameter differences P_i^t within the compressed package are crucial in preserving privacy during data exchange. The risk of disclosing specific values is minimized by converting model parameter differences into finite-bit integer representations through quantization operations. This quantization process incorporates the random constant $\mathbf{PRN}_r$ and the global model parameters from the previous round, ensuring randomness and variance in the quantization results. As this compressed package solely contains quantized results of parameter differences and does not disclose the original model parameters or user data, the privacy of individual users is well-preserved. Furthermore, the quantization operations and compression processes further mitigate the risk of information leakage.

Compared to a baseline approach, FedAvg and the similar framework Chain-PPFL, our Q-Chain FL scheme provides stronger privacy preservation through quantization operations, parameter difference transmission, and randomness preservation.

4.2 Analysis of Latency

In the previous federated learning schemes, the latency of each round of model training can be divided into three parts. T_{local} denotes the latency of the participant to calculate the local model parameters, T_{global} denotes the latency of the server to calculate the global model parameters, and $T_{\text{comm, up}}$ denotes the upload update local communication delay. When comparing the latency of the various schemes, Q-Chain FL does not take into account the time it takes for the server to broadcast the global model to the local nodes, as this time is the same for all local users. To facilitate the analysis and comparison, within the context of FedAvg, the Q-Chain FL assumes that for each round, all users have identical values for T_{local} (the latency for local model parameter computation) and $T_{\text{comm, up}}$ (the local communication delay for uploading updates).

In FedAvg, participating users upload local updates to the server in parallel, resulting in a total latency per round as follows.

$$T_{\text{total}} = T_{\text{local}} + T_{\text{global}} + T_{\text{comm, up}}. \tag{8}$$

In Chain-PPFL, the communication delay $T_{\text{comm, c}}$ includes two parts: the communication delay for uploading local model parameters (the same method as above), and the communication delay between local nodes $T_{\text{comm,sa}}$. For the sake of analysis and comparison, we assume that the local user has the same communication delay as the next user in the user list, denoted as $T_{\text{comm, nb}}$. Therefore, $T_{\text{comm, c}}$ can be computed by the equation (9).

$$T_{\text{comm, c}} = T_{\text{comm, sa}} + T_{\text{comm, up}} = (K-1) \times T_{\text{comm, nb}} + T_{\text{comm, up}}, \tag{9}$$

where K represents the number of local nodes in each round of model training. The server node needs to calculate the sum of the parameter differences and average the accumulated results. As a result, the latency $T_{\text{local, c}}$ to compute the local model

parameters is slightly larger than FedAvg, whereas the delay $T_{\text{global, c}}$ for computing global model parameters is smaller than T_{global}. Hence, the total delay of the Chain-PPFL scheme can be obtained by Equation(10).

$$T_{\text{total, c}} = T_{\text{local}} + T_{\text{global, c}} + T_{\text{comm, c}}. \tag{10}$$

In Q-Chain FL, the communication delay $T_{\text{comm, q}}$ contains two components. The first is the communication delay $T_{\text{comm, qc}}$ between local nodes, representing the time required to transmit compressed model parameter differences between nodes. The second is the communication delay $T_{\text{comm, up2}}$ for uploading the local model parameters. Similarly, we assume that the local participant has the same communication delay as other participants in the user list in the Q-Chain FL scheme, denoted as $T_{\text{comm,k}}$. Therefore, $T_{\text{comm, q}}$ can be computed by the equation (11).

$$T_{\text{comm, q}} = T_{\text{comm, qc}} + T_{\text{comm, up2}} = (K-1) \times T_{\text{comm, k}} + T_{\text{comm, up2}}, \tag{11}$$

where K represents the number of participants for this model training. In Q-Chain FL, local nodes communicate by transmitting compressed model parameter differences, which reduces the storage space and the transmission time compared to transmitting complete model parameters. After the last user on the user list uploads its local model parameters, the server node must compute the sum of parameter differences, perform dequantization, and average the accumulated results.

As a result, the delay $T_{\text{local, q}}$ for calculating the local model parameters is slightly smaller than the delay for FedAvg, the communication delay $T_{\text{comm, q}}$ between the local nodes is slightly smaller than the delay for uploading the local model parameters, and the communication delay for uploading the local model parameters $T_{\text{comm, up2}}$ is slightly smaller than $T_{\text{comm, up}}$, and the delay $T_{\text{global, q}}$ for computing global model parameters is slightly larger than T_{global}. Therefore, the total delay of the Q-Chain FL scheme can be expressed as the equation (12).

$$T_{\text{total, q}} = T_{\text{local, q}} + T_{\text{global, q}} + T_{\text{comm, q}}. \tag{12}$$

For the time calculations, the models are assumed to have the same baseline training time, we observe that $T_{\text{local, q}} \ll T_{\text{local}}$, $T_{\text{comm, qc}} \ll T_{\text{comm, sa}}$, $T_{\text{global}} < T_{\text{global, q}}$, and $T_{\text{comm, up2}} \ll T_{\text{comm, up}}$.Then, the analysis results of the above three schemes are as follows

$$T_{\text{total}} < T_{\text{total, c}} < T_{\text{total, q}}. \tag{13}$$

Considering the security requirements of the Q-Chain FL scheme, it is necessary to ensure that $T_{\text{total, q}}$ is larger than $T_{\text{local}} + (K-1) \times T_{\text{comm, k}} + T_{\text{comm, up}}$, where K represents the number of participants for this model training. Using quantization compression techniques, Q-Chain FL significantly reduces the communication latency $T_{\text{comm, k}}$ between local nodes and their neighboring nodes while performing effectively in some environments with high network bandwidth and low network latency. In addition, the scheme successfully reduces the dependence on user node devices and communication systems.

Scheme	Local Latency	Global Latency	Communication Latency
FedAvg	T_{local}	T_{global}	$T_{\text{comm, up}}$
Chain-PPFL	$T_{\text{local, c}}$ ($\uparrow$)	$T_{\text{global, c}}$ ($\downarrow$)	$(K-1) \times T_{\text{comm, nb}} + T_{\text{comm, up}}$
Q-Chain FL	$T_{\text{local, q}}$ ($\downarrow$)	$T_{\text{global, q}}$ ($\uparrow$)	$(K-1) \times T_{\text{comm, k}} + T_{\text{comm, up2}}$

Table 3. Comparison of Latency in Different Federated Learning Schemes

Table 3 summarizes the above analysis.

5 Experiments

5.1 Setup

This subsection evaluates the proposed Q-Chain FL scheme by comparing it with the FedAVG and Chain-PPFL algorithms. We implement all schemes in the Python programming language and the PyTorch framework. The evaluation is performed on three widely applied datasets: MNIST, CIFAR-10, and CelebA. For consistency, all participating clients are initialized with identical model parameters. The experimental setup is described as follows:

- **MNIST dataset**: The MNIST dataset is a benchmark for handwritten digit recognition, containing grayscale images of digits ranging from 0 to 9. Each image has a resolution of 28×28 pixels. It contains 60,000 training samples and 10,000 test samples.

Table 4. The Parameters of Experiment

CPU	AMD Ryzen5-5600@3.50 GHz
memory	16.0 GB RAM
operating system	Window 10
Learning rate	0.01
Fraction of users	0.1
batch size	10
local epochs	5

- **CIFAR-10 dataset**: The CIFAR-10 dataset is widely used in image classification tasks. It includes 50,000 training samples and 10,000 test samples across 10 object categories, and each image is a color image with 32×32.
- **CelebA dataset**: The CelebA dataset is a large-scale face attribute dataset frequently used for face recognition and attribute prediction. It contains more than 200,000 celebrity images annotated with 40 attribute labels. All images are resized to 224×224 pixels for uniformity.

2) Data segmentation: In the IID case, the Q-Chain FL shuffles the MNIST and CIFAR-10 datasets. This ensures that the samples are randomized and evenly distributed among users or training batches. However, a different approach is used for the MNIST dataset in the non-IID case. The MNIST dataset is divided into 200 segments, each containing 300 samples. This division allows for creating subsets with different characteristics, which helps simulate non-IID distributions.

3) Training Models: In this study, we employed a diverse set of models for training, including MLP, CNN, AlexNet, and ResNet-18. Each model is chosen based on its suitability for specific datasets and tasks. The details of these architectures are outlined below:

- **MLP**: The Multi-Layer Perceptron (MLP) architecture in our experiments contains two fully connected hidden layers, each containing 200 units. The ReLU (Rectified Linear Unit) activation function is used for these hidden layers to introduce non-linearity, enhancing the network's ability to capture complex patterns.
- **CNN**: The Convolutional Neural Network (CNN) architecture comprises two convolutional layers, each utilizing 5×5 kernels to extract spatial features. These layers are followed by a fully connected layer with 512 units and ReLU activation. The output layer adopts a softmax activation function to perform classification.
- **AlexNet**: AlexNet is a deep CNN designed for large-scale image classification. It includes a feature extraction module with multiple convolutional layers and ReLU activation functions, followed by a classification module with fully connected layers. The network has a total of 23,272,266 parameters.
- **ResNet-18**: ResNet-18 is a deep residual network featuring 18 layers, specifically designed to solve the vanishing gradient problem in deep networks. It is particularly effective for image classification tasks, leveraging a modular structure composed of convolutional blocks, batch normalization, and ReLU activation functions.

Table 4 shows the detailed experimental parameter settings.

5.2 Experimental Results

This section evaluates the proposed Q-Chain FL scheme from model accuracy and system overhead. For all experiments, the quantization value in our approach is set to 16 bits.

1) Accuracy Evaluation:

We assess the performance of the Q-Chain FL scheme on classification accuracy through a comprehensive set of experiments conducted on three datasets, i.e., MNIST, CIFAR-10, and CelebA. The results are compared against two baseline methods: FedAvg and Chain-PPFL. The evaluation considered both IID and non-IID data distributions across multiple experimental scenarios. The experimental results, as depicted in Figure 3, demonstrate that Q-Chain FL achieves comparable or slightly higher model accuracy than FedAvg and Chain-PPFL under IID scenarios. This parity in performance is due to the absence of noise in Q-Chain FL, enabling stable training dynamics akin to FedAvg. Moreover, in Figure 3(f), Q-Chain FL converges faster than the Chain-PPFL, attributed to its quantization mechanism, which reduces the size of model update parameters and facilitates more frequent participant updates.

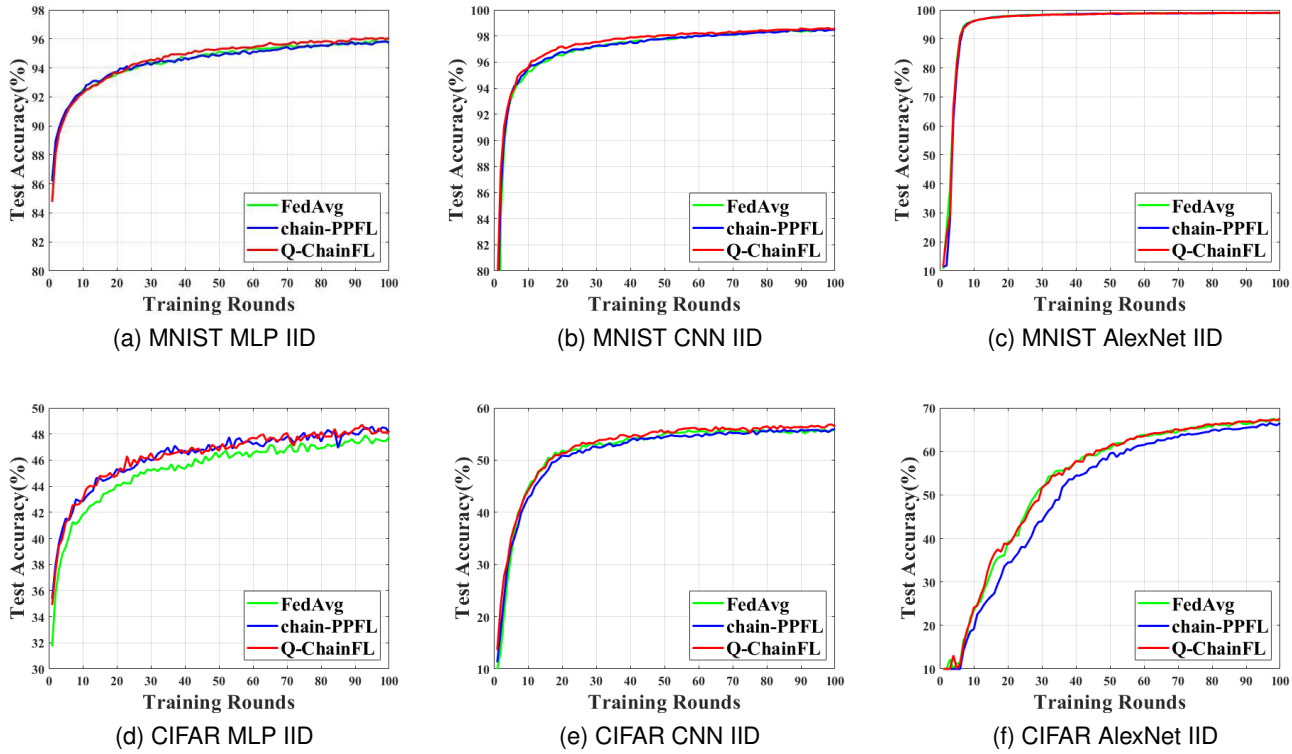


Figure 3. Accuracy comparison of Q-Chain FL, FedAvg, and Chain-PPFL on MNIST and CIFAR-10 under IID data distribution.

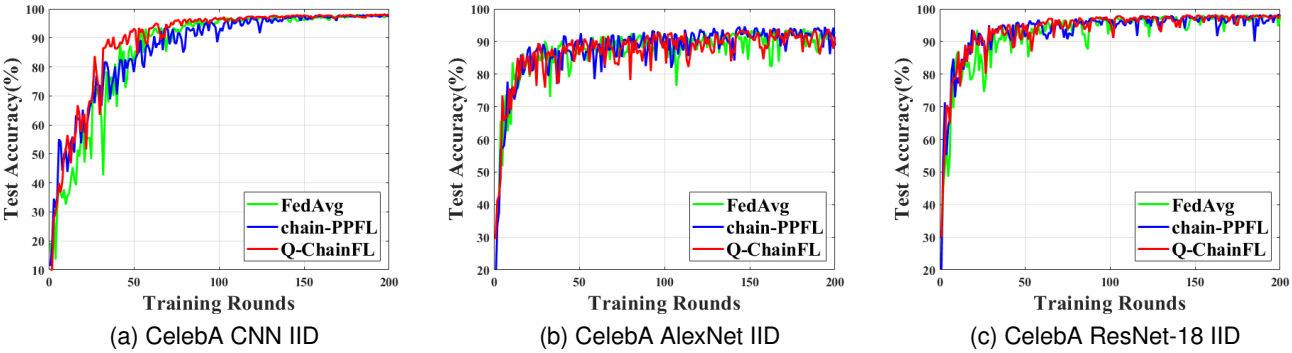


Figure 4. Accuracy comparison of Q-Chain FL, FedAvg, and Chain-PPFL on the CelebA dataset under IID conditions.

Furthermore, as shown in Figure 4, we evaluated the model accuracy convergence for CelebA under IID scenarios using three models: CNN, AlexNet, and ResNet-18. Experimental results indicate that the Q-Chain FL scheme achieves optimal accuracy and convergence speed comparable to FedAvg methods. When employing the ResNet-18 model, the scalability and robustness of Q-Chain FL in handling deeper networks are further validated.

Table 5 shows the number of communication rounds required for three federated learning algorithms, FedAvg, Chain-PPFL, and Q-Chain FL, to reach different target accuracies on the CIFAR dataset (independent and identically distributed, using the AlexNet model). The target accuracies are 45%, 55%, and 65%, respectively. At each target accuracy, Q-Chain FL requires fewer communication rounds than FedAvg and Chain-PPFL, indicating that Q-Chain FL converges faster under this dataset and model setting.

In the case of non-IID data, applying quantization compression to chain-structured federated learning enables the model to reach the same accuracy within fewer communication rounds and computation overhead, as shown in Table 6.

In Figure 5, on the MNIST and CIFAR, the variation of test accuracy under different numbers of training epochs is observed.

Table 5. CIFAR AlexNet IID

Target Accuracy	FedAvg	Chain-PPFL	Q-Chain FL
45%	26	30	25
55%	36	40	35
65%	73	80	71

Table 6. MNIST non-IID

Model-Accuracy	FedAvg	Chain-PPFL	Q-Chain FL	Computation Overhead
MLP-93%	1.5x	1.5x	1.0x	9.6×10^5 ms
CNN-98%	1.7x	1.7x	1.4x	9.3×10^5 ms
AlexNet-97%	1.3x	1.3x	1.0x	8.9×10^5 ms
				(magnitude:100 rounds)

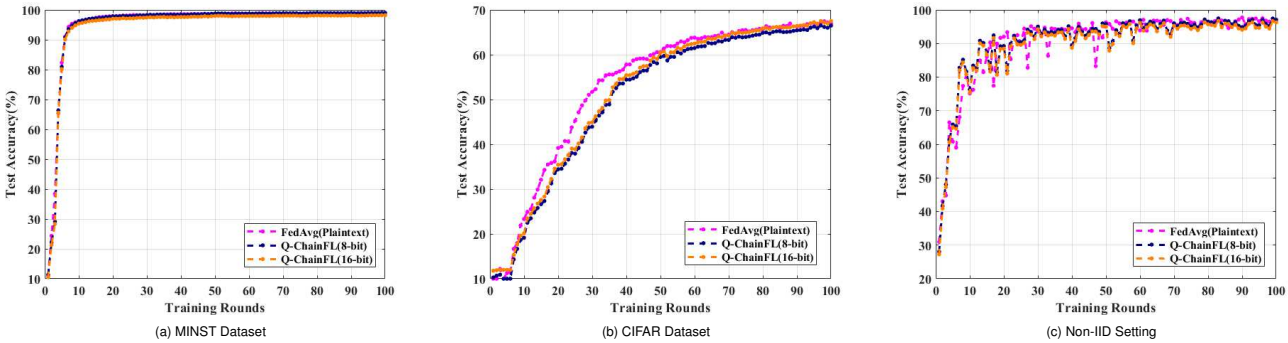


Figure 5. Quantization Experiments of Federated Learning in Diverse Datasets and Distributions.

In the legend of the experimental results, "Plaintext FL" refers to the plaintext baseline of FL with model updates in 32-bit floating-point precision, and "8/16-bit quantization" means that the values are quantized into 8-bit or 16-bit integers during model updates to enhance the communication efficiency in the ciphertext domain. The experimental results show that the accuracy of the security protocol is close to that of the plaintext baseline and can achieve a comparable test accuracy, and the quantized integer representation with an appropriate bit width has no significant negative impact on the training quality of the model.

2) Performance Evaluation.

Table 7 summarizes the impact of various transmission settings on uplink communication overhead and model accuracy for MLP, CNN, AlexNet, and ResNet-18 models across MNIST, CIFAR-10, and CelebA datasets. The results demonstrate that quantized transmission significantly reduces communication overhead compared to plain 32-bit transmission while maintaining model accuracy. For the MLP model, uplink communication overhead for MNIST decreases from 7.96GB under plain transmission to 1.99GB with 8-bit quantization and 3.98GB with 16-bit quantization, while accuracy remains at 0.95 for MNIST and 0.45 for CIFAR-10. Similar trends are observed for CNN. For CNN, plain transmission demands 2.28GB when dealing with the MNIST, but 8-bit and 16-bit quantization transmissions require 0.57GB and 1.14GB, respectively, without affecting the accuracy (0.98 for MNIST and 0.55 for CIFAR-10).

For AlexNet, the uplink communication overhead is the highest, with plain transmission demanding 93.2GB for CIFAR-10, reduced to 23.3GB and 46.6GB under 8-bit and 16-bit quantization transmissions, respectively, while maintaining accuracy at 0.68. When applying ResNet-18 on CelebA, plain transmission requires 140GB for uplink communication, while 8-bit and 16-bit quantization transmissions significantly reduce the overhead to 35GB and 70GB, respectively, maintaining an accuracy of 0.79. These results confirm that quantized transmission effectively reduces communication overhead while preserving accuracy, demonstrating its scalability and efficiency in federated learning scenarios, especially for complex models such as AlexNet and ResNet-18.

Table 7. Uplink Communication Overhead and Model Accuracy under Different Transmission Settings

Model	Transmission Type	Quantization	Uplink Communication Overhead	Accuracy
MLP	Plain Transmission (32-bit)	-	7.96GB	0.95 (MNIST) / 0.45 (CIFAR-10)
	Quantized Transmission	8-bit	1.99GB	0.95 (MNIST) / 0.45 (CIFAR-10)
	Quantized Transmission	16-bit	3.98GB	0.95 (MNIST) / 0.45 (CIFAR-10)
CNN	Plain Transmission (32-bit)	-	2.28GB	0.98 (MNIST) / 0.55 (CIFAR-10)
	Quantized Transmission	8-bit	0.57GB	0.98 (MNIST) / 0.55 (CIFAR-10)
	Quantized Transmission	16-bit	1.14GB	0.98 (MNIST) / 0.55 (CIFAR-10)
AlexNet	Plain Transmission (32-bit)	-	93.2GB	- / 0.68 (CIFAR-10)
	Quantized Transmission	8-bit	23.3GB	- / 0.68 (CIFAR-10)
	Quantized Transmission	16-bit	46.6GB	- / 0.68 (CIFAR-10)
ResNet-18	Plain Transmission (32-bit)	-	140GB	- / 0.79 (CelebA)
	Quantized Transmission	8-bit	35GB	- / 0.79 (CelebA)
	Quantized Transmission	16-bit	70GB	- / 0.79 (CelebA)

Figure 6 compares the accuracy of the global models of three algorithms, FedAvg, FedProx, and Q-Chain FL, under different numbers of clients on the MINST and CIFAR datasets. In both the MINST and CIFAR datasets, as the number of clients increases, the accuracy of all three algorithms rises, and Q-Chain FL consistently exhibits the highest accuracy.

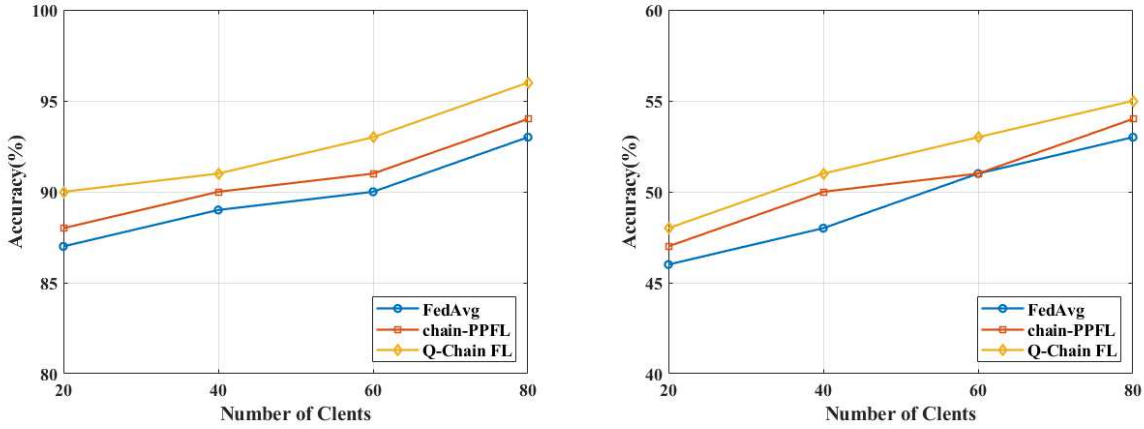


Figure 6. Compare the accuracy of the global model for different numbers of clients.

Table 8 presents the communication overhead when applying CNN on the CelebA dataset, under different numbers of clients, transmission modes, and quantization levels. As the number of clients increases from 20 to 80, the total communication overhead rises significantly for both plain 32-bit transmission and quantized transmission. For instance, in plain 32-bit transmission, the total overhead for 20 clients is 32 GB, and for 80 clients, it reaches 148 GB. Quantized transmission shows a remarkable advantage in reducing overhead. Taking 20 clients as an example, the total overhead for 8-bit quantization is 8.5 GB, and for 16-bit quantization is 17 GB, much lower than that of plain transmission. Across scenarios with different numbers of clients, the effectiveness of quantization technology in reducing communication overhead is prominent. In plain 32-bit transmission, the average overhead per client remains relatively stable. Quantized transmission has a lower and stable average overhead, fully demonstrating its effectiveness and stability in federated learning scenarios with varying scales of client participation.

3) Security Evaluation. In the privacy leakage comparison experiment, a matching operation is performed on all parameter gradients of the training model in FL, and the experiment adopts randomly initialized weights. Here, the gradient matching loss can represent the discriminability of the original samples. Specifically, the smaller the matching loss corresponding to the gradient value, the greater the information leakage. When the value of the matching loss function exceeds 0.15, it can be determined that there is no information leakage. Figure. 7 presents the experimental results of the anti-leakage effectiveness. It can be seen from it that the FedAVG scheme fails to effectively prevent the leakage of sensitive information, while the Q-Chain FL successfully achieves privacy preservation and ensures information security during the model training process.

Table 8. Communication Overhead for Different Numbers of Clients Using CNN on CelebA

Number of Clients	Transmission Mode	Quantization Level	Total Communication Overhead	Average Communication Overhead per Client
20	Plain Transmission (32-bit)	-	32GB	1.6GB
20	Quantized	8-bit	8.5GB	0.43GB
20	Quantized	16-bit	17GB	0.85GB
40	Plain Transmission (32-bit)	-	74GB	1.85GB
40	Quantized	8-bit	18GB	0.45GB
40	Quantized	16-bit	36GB	0.9GB
60	Plain Transmission (32-bit)	-	112GB	1.87GB
60	Quantized	8-bit	35GB	0.58GB
60	Quantized	16-bit	69GB	1.15GB
80	Plain Transmission (32-bit)	-	148GB	1.85GB
80	Quantized	8-bit	38GB	0.48GB
80	Quantized	16-bit	76GB	0.95GB

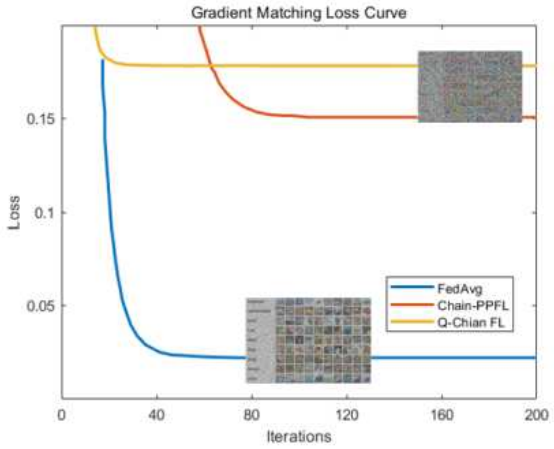


Figure 7. The experimental results of the anti-leakage effectiveness.

Table 9 presents a comprehensive analysis of the performance of three federated learning algorithms. The analysis is centered around the model accuracy under varying proportions of malicious participants within the MNIST and CIFAR-10 datasets, both of which follow an IID. For the MNIST (IID) dataset, when the proportion of malicious participants is 0%, FedAvg attains an accuracy of 97.6, FedProx reaches 97.9, and Q-Chain FL achieves the highest accuracy of 98.1. As the proportion of malicious participants increases to 10%, 20%, and 30%, the accuracies of all three algorithms decline. However, Q-Chain FL consistently maintains the highest accuracy at each malicious participation level, with values of 82.4, 75.2, and 68.1, respectively. Regarding the CIFAR-10 (IID) dataset, at 0% malicious participants, FedAvg has an accuracy of 52.0, FedProx has 53.5, and Q-Chain FL has the highest accuracy of 55.0. Similarly to the MNIST scenario, as the percentage of malicious participants increases, the accuracy of all algorithms decreases. Once again, Q-Chain FL outperforms the other two algorithms, with accuracies of 48.0, 42.0, and 35.0 at 10%, 20%, and 30% malicious participation, respectively.

Table 9. Model Accuracy of Different Federated Learning Algorithms under Varying Proportions of Malicious Participants on MNIST and CIFAR-10 Datasets with IID Distributions

Malicious Participant Ratio	**MNIST (IID)**			**CIFAR-10 (IID)**		
	FedAvg	**FedProx**	**Q-Chain FL**	**FedAvg**	**FedProx**	**Q-Chain FL**
0%	97.6	97.9	**98.1**	52.0	53.5	**55.0**
10%	52.3	72.1	**82.4**	35.0	42.0	**48.0**
20%	38.7	60.5	**75.2**	25.0	33.0	**42.0**
30%	25.6	45.3	**68.1**	18.0	25.0	**35.0**

The results of the Q-Chain FL show that the quantization compression technique in chain-structured federated learning not only enhances the efficiency of the system but also strengthens the privacy preservation of the data and models. Thus, federated learning exhibits superior performance in the face of large-scale and privacy-sensitive data.

6 Conclusion

This paper presents an innovative framework for federated learning, Q-Chain FL, designed to address the critical challenges of communication efficiency and privacy preservation inherent in traditional FedAvg-based schemes. By seamlessly integrating quantization compression techniques into an advanced chained architecture, Q-Chain FL substantially reduces communication and storage overhead while significantly accelerating the convergence of global models. To thoroughly evaluate its performance and robustness, we conduct extensive experiments on diverse datasets, including MNIST, CIFAR-10, and CelebA, leveraging multiple model architectures such as convolutional neural networks (CNNs) and ResNet-18. These results highlight that Q-Chain FL demonstrates superior efficiency, scalability, and robustness, while consistently maintaining or enhancing model accuracy across various data distributions, particularly in large-scale, heterogeneous, constrained, and privacy-sensitive environments.

Although Q-Chain FL shows significant promise, challenges remain in adapting to the dynamic nature of real-world federated learning scenarios. Future research will prioritize improving mechanisms for client participation and withdrawal, as well as developing dynamic management strategies to optimize the framework's adaptability in environments characterized by non-IID data distributions and resource heterogeneity.

Acknowledgements

The research is financially supported by the Shanghai Natural Science Foundation (grant numbers 22ZR1481000 and 20ZR1419700) and the Lion Rock Labs of Cyberspace Security (grant number LRL24017).

Data availability statement

MNIST: http://yann.lecun.com/exdb/mnist/.
CelebA: http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html.
CIFAR-10: https://www.cs.toronto.edu/~kriz/cifar.html.

References

1. Ge, Z., Song, Z., Ding, S. X. & Huang, B. Data mining and analytics in the process industry: The role of machine learning. *IEEE Access* **5**, 20590–20616 (2017).
2. Alam, M. *et al.* Machine learning in cybersecurity: Opportunities and challenges. In *2024 IEEE 16th International Conference on Computational Intelligence and Communication Networks (CICN)*, 663–670 (2024).
3. Lone, A. N., Mustajab, S. & Alam, M. A comprehensive study on cybersecurity challenges and opportunities in the iot world. *Secur. Priv.* **6**, e318 (2023).
4. Alam, M. & Mustaqeem, M. Reinforcing defect prediction: a reinforcement learning approach to mitigate class imbalance in software defect prediction. *Iran J. Comput. Sci.* **8**, 151–162 (2025).
5. McMahan, H. B., Moore, E., Ramage, D. & y Arcas, B. A. Federated learning of deep networks using model averaging. *arXiv,preprint* **1602.05629** (2016).
6. Konečný, J. *et al.* Federated learning: Strategies for improving communication efficiency. *arXiv,preprint* **1610.05492** (2016).
7. Zhao, C. *et al.* Secure multi-party computation: Theory, practice and applications. *Inf. ences* **476** (2018).
8. Yang, Q., Liu, Y., Chen, T. & Tong, Y. Federated machine learning: Concept and applications. *arXiv,preprint* **1902.04885** (2019).
9. McMahan, B., Moore, E., Ramage, D., Hampson, S. & y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In Singh, A. & Zhu, X. J. (eds.) *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, vol. 54 of *Proceedings of Machine Learning Research*, 1273–1282 (PMLR, 2017).
10. Kairouz, P., McMahan, H. B. & et al, B. A. Advances and open problems in federated learning. *Found. Trends Mach. Learn.* **14**, 1–210 (2021).

11. Deepak, Upadhyay, M. K. & Alam, M. Edge computing: Architecture, application, opportunities, and challenges. In *2023 3rd International Conference on Technological Advancements in Computational Sciences (ICTACS)*, 695–702 (2023).

12. Paragliola, A. & Giovanni. Evaluation of the trade-off between performance and communication costs in federated learning scenario. *Futur. Gener. Comput. Syst.* **136**, 282–293 (2022).

13. Casalicchio, B., Emiliano *et al.* Flwb: a workbench platform for performance evaluation of federated learning algorithms. In *2023 IEEE International Workshop on Technologies for Defense and Security (TechDefense)*, 401–405 (2023).

14. Yang, C., Guang *et al.* Ringfed: Reducing communication costs in federated learning on non-iid data (2021).

15. Wang, D., Luping *et al.* Cmfl: Mitigating communication overhead for federated learning. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, 954–964 (2019).

16. Al-Saedi, E., A., A. *et al.* Fedco: Communication-efficient federated learning via clustering optimization. *Futur. Internet* (2022).

17. Mothukuri, V. *et al.* A survey on security and privacy of federated learning. *Futur. Gener. Comput. Syst.* **115**, 619–640 (2021).

18. Li, Y. *et al.* Privacy-preserving federated learning framework based on chained secure multiparty computing. *IEEE Internet Things J.* **8**, 6178–6186 (2021).

19. Li, T., Sahu, A. K., Talwalkar, A. & Smith, V. Federated learning: Challenges, methods, and future directions. *arXiv,preprint* **1908.07873** (2019).

20. Rahman, S. A., Tout, H., Mourad, A. & Talhi, C. Fedmccs: Multicriteria client selection model for optimal iot federated learning. *IEEE Internet Things J.* **8**, 4723–4735 (2021).

21. Huang, T., Lin, W., Shen, L., Li, K. & Zomaya, A. Y. Stochastic client selection for federated learning with volatile clients. *IEEE Internet Things J.* **9**, 20055–20070 (2022).

22. Wang, X. *et al.* Attention-weighted federated deep reinforcement learning for device-to-device assisted heterogeneous collaborative edge caching. *IEEE J. Sel. Areas Commun.* **39**, 154–169 (2021).

23. Li, S., Ngai, E. C. H., Ye, F. & Voigt, T. Auto-weighted robust federated learning with corrupted data sources. *ACM Trans. Intell. Syst. Technol.* **13**, 73:1–73:20 (2022).

24. Zhou, S. *et al.* Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv,preprint* **1606.06160** (2016).

25. Courbariaux, M., Bengio, Y. & David, J. Binaryconnect: Training deep neural networks with binary weights during propagations. *arXiv,preprint* **1511.00363** (2015).

26. Ullrich, K., Meeds, E. & Welling, M. Soft weight-sharing for neural network compression. *arXiv,preprint* **1702.04008** (2017).

27. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R. & Bengio, Y. Quantized neural networks: Training neural networks with low precision weights and activations. *J. Mach. Learn. Res.* **18**, 187:1–187:30 (2017).