HiQuFlexAsync: Hierarchical Federated Learning with Quantization, Flexible Client Selection and Asynchronous Communication

Ze Zhao [1], Yifan Liu [1](✉), Donglin Pan [1], Yi Liu [2](✉), Xiaofei Li [2], Zhenpeng Liu [1,2]

[1] School of Cyberspace Security and Computer, Hebei University, Baoding, China
`lyf@hbu.edu.cn(Yifan Liu)`

[2] Information Technology Center, Hebei University, Baoding, China
`liuyi@hbu.edu.cn(Yi Liu)`

Abstract. In three-tier federated learning at the cloud-edge, uneven data distribution can lead to a decrease in model performance, while the increased communication demands of multi-tier federated learning may impact system efficiency. To mitigate the impact of these challenges on federated learning performance, a novel method named HiQuFlexAsync has been introduced. HiQuFlexAsync is an innovative asynchronous three-tier federated learning approach with quantization capabilities. Within the framework of HiQuFlexAsync, a new quantizer is employed to naturally compress local and edge gradients, and an algorithm called "Cost-Optimized Heterogeneous Client-Edge Association" (COHEA) is developed. This algorithm aims to optimize the client selection process for federated learning by leveraging data heterogeneity and the physical diversity of clients. Simulation experiments on the MNIST and CIFAR-10 datasets demonstrate that compared to the traditional three-tier architecture HierFAVG, HiQuFlexAsync achieves an approximate 5.6% increase in accuracy and a 12.2% enhancement in efficiency.

Keywords: Federated Learning, Hierarchical Mechanism, Quantization, Client Selection, Asynchronous Aggregation.

1 Introduction

In the era of information, data plays a pivotal role. However, traditional data training methods are plagued by issues such as privacy breaches, centralization, and computational resource constraints [1]. To address these challenges, federated learning has emerged as a distributed machine learning paradigm of interest, capable of conducting model training across devices, regions, and institutions while safeguarding data privacy. Yet, the physical and data heterogeneity among participants may diminish the efficacy of model training [2].

Scholars have emphasized the crucial role of selecting participants with similar device capabilities and data distributions in federated learning. This aids in mitigating instability during model training and enhancing global convergence. Furthermore, in the context of large-scale models and numerous participants, they have proposed the concept of introducing a three-tier architecture encompassing cloud, edge, and endpoint. This approach aims to alleviate cloud computing resource constraints, while concurrently improving communication efficiency and model training effectiveness [3]. In addition, synchronous training in federated learning is plagued by significant communication overhead, prolonged training durations, and diminished system robustness. However, the adoption of a hybrid synchronous and asynchronous training paradigm within a three-tier federated learning framework holds promise in mitigating these challenges, thereby facilitating more efficient and secure model training. Furthermore, by quantizing models, it is possible to reduce communication overhead and enhance the efficiency of federated learning.

This paper introduces HiQuFlexAsync, an innovative asynchronous three-tier federated learning solution. The main contributions of this paper are as follows:

- An innovative approach called HiQuFlexAsync has been proposed, which is a novel asynchronous three-tier federated learning method with quantization capability. In the process of aggregating and updating the model, a novel quantizer is introduced, enabling natural compression and quantization of local and edge gradients. Quantizing gradients helps reduce communication rounds, thereby enhancing the efficiency of the federated learning process.
- An algorithm named COHEA is introduced, addressing the client selection problem in federated learning. This algorithm comprehensively considers communication latency, computational delay, and the similarity of data distribution. Through a cost evaluation formula, it optimizes the selection of clients, enabling federated learning to achieve better performance in heterogeneous environments.
- Simulation experiments conducted on the MNIST and CIFAR-10 datasets demonstrate that HiQuFlexAsync, compared to the traditional three-tier architecture HierFAVG, achieves approximately a 5.6% increase in accuracy and about a 12.2% improvement in efficiency.

The remaining sections of this paper are organized as follows: Section 2 provides an overview of relevant work in federated learning. In Section 3, the process of gradient quantization, the HiQuFlexAsync framework, and the client selection algorithm COHEA are detailed. Section 4 presents the conducted simulation experiments and the analysis of their results. Finally, Section 5 concludes the paper.

2 Related Work

2.1 Client Selection

Nishio et al. [4] have proposed the FedCS protocol, which leverages a greedy algorithm to maximize the inclusion of eligible clients. Wang et al. [5] proposed the Acct framework, which ingeniously combines change detection with multi-armed bandit techniques. Zheng et al. [6] have proposed a heuristic algorithm based on an energy-accuracy balance, aimed at optimizing client selection to achieve a balance between energy consumption and learning accuracy.

2.2 Gradient Quantization

Gradient quantization technology compresses and approximates parameter gradients in deep neural networks to reduce computational and communication costs while maintaining model performance, which has been widely applied across various domains. For instance, Chen et al. [7] successfully applied quantization techniques to edge intelligent vehicle networks to reduce training computational complexity and achieve efficient training. Lu et al. [8] leveraged quantization technology in the field of computer vision, yielding significant results. Furthermore, Dupuy et al. [9] utilized gradient quantization in natural language processing, achieving favorable outcomes.

2.3 Layered Federated System

The three-tier architecture has found extensive application within federated learning paradigms. Luo et al. [10] proposed the HFEL framework, which achieves low latency and high energy efficiency in machine learning by performing partial model aggregation on edge servers. Lim et al. [11] introduced a layered federated learning framework tailored for edge intelligence environments. Chen et al. [12] adopted the TP-DDPG framework to learn multiple decisions using a DDPG-based approach, introducing a new algorithm for client association and bandwidth allocation, taking into account the issue of client procrastination.

3 System Model

To alleviate the impact of device heterogeneity and network latency on federated learning performance, a new scheme utilizing a cloud-edge-end three-tier architecture for federated learning, named HiQuFlexAsync, is proposed. This section details the proposed system model for the cloud-edge-end federated learning architecture. The entire system model is illustrated in Figure 1.

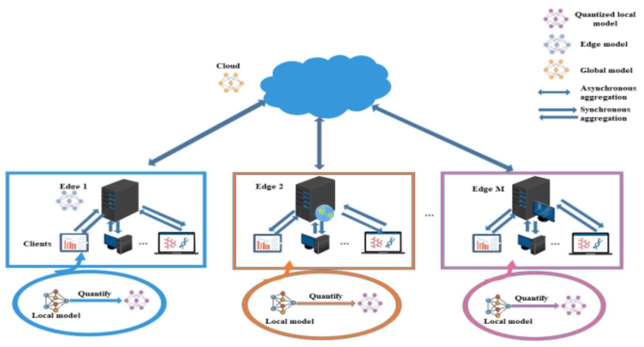


Fig. 1. Overview of System Mode.

Table 1 displays the key symbols and their meanings used in this paper.

Table 1. Key Notations

Symbols	Definition
C_i	Client i
E_j	Edge Sever j
S	Set of clients connected to an edge server
N	Number of clients
M	Number of Edge Severs
D_i	Dataset on client i
g_i	Total number of CPU cycles required for executing dataset D_i
B_i	Bandwidth capacity of client i
c	Transmission power density
h_i	Channel gain
N_0	Background noise
S_i	Size of the training model of C_i
$d_{i,j}$	Distance from client C_i to edge server E_j
$w_i(t)$	Local parameters of client i
$w_j(t)$	Local parameters of edge server j
w_t	Global parameters

3.1 Gradient Quantization

Gradient quantization, a technique aimed at compressing single-precision floating-point numbers into finite bits, mitigates transmission bandwidth demands and accelerates the SGD training process of deep neural networks. By encoding and quantizing gradient parameters after each descent iteration and transmitting them to subsequent nodes, this method significantly boosts training efficiency. In our study, a method was devised where local clients transmit quantized model parameters to edge servers. On the edge servers, the quantized model parameters are decoded and aggregated. Subsequently, the aggregated edge parameters are re-quantized and sent to the cloud. At the cloud, decoding is performed, enabling further global model aggregation. Figure 2 illustrates this process. This approach of quantized model updates helps reduce communication overhead and safeguard data privacy.

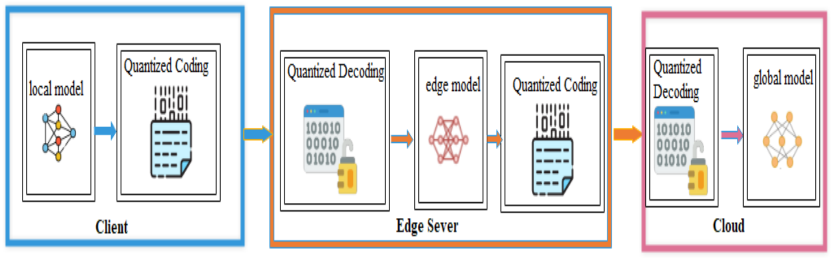


Fig. 2. Transmission framework for quantized stochastic gradient descent.

A low-precision quantizer tailored for natural compression of parameters was selected for gradient quantization. The definition of this quantizer is as follows:

$$Q(w_k) = \|w\|_2 \operatorname{sgn}(w_k) C_k(w_k) \tag{1}$$

Where

$$\operatorname{sgn}(w_k) = \begin{cases} 1, & w_k > 0 \\ -1, & w_k \leq 0 \end{cases} \tag{2}$$

And

$$C_k(w_k) = \frac{1}{1+e^{-|w_k-\mu|/\sigma}} \tag{3}$$

The sign function, $\operatorname{sgn}(w_k)$, determines the quantized symbol based on the positivity or negativity of the weight w_k . Aimed at minimizing quantization error, the adaptive quantization coefficient $C_k(w_k)$ is defined in relation to the gradient's sensitivity and distribution, with μ and σ representing the mean and standard deviation of the gradient distribution, respectively. This function is designed to offer higher quantization precision for gradient values near the mean, while allowing coarser quantization for values far from the mean, thereby balancing efficiency and accuracy.

3.2 The processs of HiQuFlexAsync

The proposed architecture comprises a set of client devices $C=\{C_1, C_2, C_3...., C_N\}$, a set of edge servers $E = \{E_1, E_2, E_3, ..., E_M\}$ (where $M<N$), and a central node located in the cloud. The entire federated learning process encompasses two pivotal aggregation update mechanisms: synchronous aggregation updates between edge servers and clients, and asynchronous aggregation updates between edge servers and the cloud. The choice of implementing synchronous aggregation updates between edge servers and clients is driven by the objective to achieve real-time and low-latency model updates. On the other hand, asynchronous aggregation updates between edge servers and the cloud are designed to fully leverage the cloud's computational resources and storage capacity for handling large-scale data and complex model updates. This division of labor effectively balances the demands for real-time processing in edge computing environments with the cloud's capability to manage large-scale data processing. The specific steps are as follows:

Step 1 The central node in the cloud initializes the model and sends the initial model to various edge servers. These edge servers then select client devices to participate in the federated learning training.

Step 2 After the edge servers select the clients for federated learning, they distribute the initial model to these clients. Upon receiving the initial model, clients conduct local training. The clients update the model using the Stochastic Gradient Descent (SGD) algorithm, where t represents the update step index, and α represents the gradient descent step index. Moreover, In order to mitigate overfitting during the local model training process, we have implemented gradient regularization as a preventive measure. The local model parameters are updated during the training process as follows:

$$w_i(t)= w_i(t - 1) - \alpha \nabla G_i(w_i(t - 1)) \tag{4}$$

Where $G_i(w_i(t)) = F_i\,(w_i(t) + \frac{\lambda}{2}\|w_i(t) - w_i(0)\|_2^2$ and $G_w = \sum_{i=1}^{n} \frac{D_i G_i(w)}{D}$, $n \in \{1, . . . ,N\}$

Step 3 Upon completion of local training, the clients quantize their updated local parameters and transmit the quantized data to the connected edge server E_j. Upon receiving the data, the edge server performs decoding followed by edge aggregation to update the edge parameters according to formula (5). In equation (5), q_j represents the number of clients managed by edge server E_j, D_i is the total number of data samples of client i, and D_j is the total number of data samples of all clients connected to E_j.

$$w_j(t)=\frac{\sum_{i=1}^{q_j} D_i Q(w_i(t))}{D_j} \tag{5}$$

Step 4 After completing the aggregation update between the edge servers and the local clients, the edge servers quantize the updated parameters and send the quantized edge parameters to the cloud. During the aggregation process between the edge servers and the cloud, an asynchronous aggregation method is used to update the model. A staleness function and a hybrid hyperparameter α_t are defined, where α_t controls the impact of the edge model on the global model. The criteria for selecting the staleness function consider the following three aspects:

- Exponential decay functions have desirable characteristics in handling time differences, effectively reflecting the impact of time intervals on model update speed.
- The positive coefficient k in the function can control the decay speed, thus flexibly adjusting the time step of model updates to accommodate different edge node latency scenarios.
- The chosen function should reasonably reflect the latency of edge nodes, and for nodes with significant delays, it should effectively reduce the speed of model updates to avoid excessive impact on the overall model aggregation process.

Therefore, the exponential decay function s(x) is chosen as the standard for the staleness function due to its reasonableness and flexibility on the time scale. This function is well-suited to meet the needs of asynchronous aggregation in federated learning.

$$s(x)= e^{(-k*x)} \tag{6}$$

In this staleness function s(x), x represents the time difference, and k is a positive coefficient controlling the decay speed. The function s(x) outputs an adjustment factor based on the delay situation, which is used to update the time step, adapting to the performance differences of edge nodes. During the model aggregation process, two formulas are used to implement the update of global parameters.

$$\alpha_t \leftarrow \alpha \times s(T - \tau) \tag{7}$$

$$w_t \leftarrow (1-\alpha_t)w_{t-1} + \alpha_t Q(w_j(t)) \tag{8}$$

Equation (7) is utilized to adjust α_t, where $T-\tau$ represents the latency in the model update of the edge nodes. By employing Equation (8), the global model parameter w_t is aggregated.

Step 5 Once the cloud receives the quantized edge parameters sent by the edge server, it begins by decoding these edge parameters before updating the global parameter wt. After the update is completed, the global parameters are sent back to the edge server that provided the edge parameters. Subsequently, the edge server disseminates the global parameters to its connected clients, who, upon receiving the updated global parameters, initiate a new round of local training.

Table 2. Algorithm 1 illustrates the procedural framework of the HiQuFlexAsync algorithm.

```
Algorithm 1: Hierarchical Asynchronous Federated Learning with Quantization
Input: learning rate $\gamma$ , number of local updates k
Output: global model parameters $w_t$
1: procedure Hierarchical Federated Flexible Asynchronous
2: Initialize: global model parameters $w_0$, max local update rounds t=0,
               local update rounds for client I, $t_i$ =0
3: use the Algorithm 2 to assign clients to edge servers
4: while t <k do
5: Client update:
6:     for each client i ∈ {1, 2, 3...,N } do
7:         Receive $w_i(t-1)$
8:         update $w_i(t)= w_i(t - 1) - \alpha\nabla G_i(w_i(t - 1))$
9:         $Q(w_i(t)) \longleftarrow ||w||_2 sgn(w_i)C_i(w_i)$
10:        send $Q(w_i(t))$ to the connected edge servers.
11:        if i received updated parameters from edge server
12:            Start a new round of local training
13:            $t_i$ +=1, and t=max($t_i$)
14:        end if
15:    end for
16: Edge sever update:
17:    for each edge node m ∈ {1, 2, 3...,M } do
18:        Receive $Q(w_i(t))$
19:        update $w_j(t)=\frac{\sum_{i=1}^{q_j} D_i Q(w_i(t))}{D_j}$
20:        $Q(w_j(t)) \longleftarrow ||w||_2 sgn(w_j)C_j(w_j)$
21:        send $(Q(w_j(t)), \tau)$ to the cloud
22:        if received updated parameters from cloud
23:            Send the $w_t$ to the connected client
24:        end if
25:    end for
26: Cloud update:
27:    receive $(Q(w_j(t)), \tau)$
28:    update $\alpha_t \leftarrow \alpha \times s(T - \tau)$
29:    update $w_t \leftarrow (1 - \alpha_t)w_{t-1} + \alpha_t Q(w_j(t))$
30:    Send $w_t$ to the corresponding edge node
31: end while
32: return the global model parameters $w_t$
```

3.3 The Cost-Optimized Heterogeneous Client-Edge Association Algorithm

Edge servers establish a designated set of connected clients, referred to as S. Initially, within their respective communication ranges, these edge servers transmit information to the clients. Upon receipt of this information, the clients are tasked with deciding whether to participate in federated learning. If a client consents, it is obliged to send its status information back to the edge server. Consequently, the edge server is enabled to select clients based on the information received from them. To initiate this process, Formula (9) is employed for the computation of the data transmission rate.

$$r_i = B\log_2\left(1+\frac{p_i h_i^2}{N_0}\right) \tag{9}$$

r_i symbolizes the data transmission rate. The bandwidth capacity of the clients, denoted as B, is assumed to be uniform across all clients. The transmission power density is indicated by p_i, channel gain by , and noise power by N_0. After calculating the data transmission rate, Formulas (10) and (11) are used to compute the client's transmission delay and propagation delay.

$$\delta_i^T = \frac{s_i}{r_i} \tag{10}$$

$$\delta_{i,j}^P = \frac{d_{i,j}}{c} \tag{11}$$

In Formula (10), s_i denotes the size of the model information being transmitted. In Formula (11), $d_{i,j}$ represents the distance from client C_i to the edge server E_i, with c signifying the speed of light. Following the computation of the client's transmission and propagation delays, Formula (12) is employed to calculate the communication delay, and Formula (13) is used to determine the client's computational delay.

$$T_{COM} = \delta_i^T + \delta_{i,j}^p \tag{12}$$

$$T_{CMP} = \frac{g_i}{k_i} \tag{13}$$

In Formula (13), g_i indicates the total number of CPU cycles required to execute D_i, and k_i denotes the CPU frequency of C_i. Having calculated the client's communication and computational delays, the total delay Ttotal is simply the sum of these two delays, as shown in Formula (14).

$$T_{total} = T_{COM} + T_{CMP} \tag{14}$$

Upon calculating the total delay for a client, we utilize the Bhattacharyya distance to measure the similarity of label distributions between the client seeking to join and those already connected to the edge server. The Bhattacharyya distance ranges from 0 to 1, where a distance of 0 indicates that the two probability distributions are identical (we use the label distribution of the client's data to represent its probability distribution). The definition of the Bhattacharyya coefficient is as follows:For discrete values, the Bhattacharyya coefficient is defined as

$$BC(p,q) = \sum \sqrt{p(x)q(x)} \tag{15}$$

,and for continuous values,it is defined as

$$BC(p,q) = \sum \sqrt{p(x)q(x)dx} \tag{16}$$

After obtaining the Bhattacharyya coefficient, the Bhattacharyya distance can be defined as

$$DB(p,q) = -\ln(BC(p,q)) \tag{17}$$

Suppose the data distribution of the client to be selected is p_1, and the average data distribution of the clients already connected to the server is q_1. In that case, the similarity between the two distributions can be calculated using the Bhattacharyya distance:

$$DB(p_1,q_1) = -\ln(BC(p_1,q_1)) \tag{18}$$

The smaller the value of $DB(p_1,q_1)$, the more similar the two data distributions are. We use Formula (19) to evaluate the cost of participating in federated learning for potential clients, obtaining their EXP (EXP refers to EXPENSE, indicating the training cost of participating in federated learning):

$$EXP_k = T_{total} + \lambda DB\left(\frac{|D_m|\cdot FD_m + |D_k|\cdot FD_k}{|D_m| + |D_k|}, FD_m\right) \tag{19}$$

In Formula (19), λ serves as a weight parameter that balances the trade-off between resource heterogeneity and data heterogeneity. FD_k is the label distribution of the current client to be selected. FDm is defined as

$$FD_m=(p_1,p_2,p_3,\ldots,p_n) \tag{20}$$

which is a distribution containing various labels and their corresponding probabilities. Here, p_i represents the occurrence probability of label i in clients connected to edge node m. Note that the length of FD_m is equal to the number of labels, and the sum of all probability values is 1. By calculating the EXP of the client to be selected, EXP_k, the smaller the value of EXP_k , the less the total delay of the newly joined client and the more similar its data distribution is to the clients already connected to the edge server. This approach not only considers the physical heterogeneity of devices but also the heterogeneity of data, reducing the impact of heterogeneity on federated learning from both aspects, thereby enhancing the efficiency of federated learning.

Having determined the EXP of the client to be selected, we can commence the client selection process. The specific strategy for edge servers to select clients is as follows:

- If the client cluster of edge node E is empty, the edge node selects the client with the least response delay from the unassociated clients within its communication range. Otherwise, the edge node selects the client k with the lowest cost EXP.
- If a client is currently selected by multiple edge nodes, the client will randomly choose one edge node.
- The bilateral selection process continues until all clients are associated with an edge node.

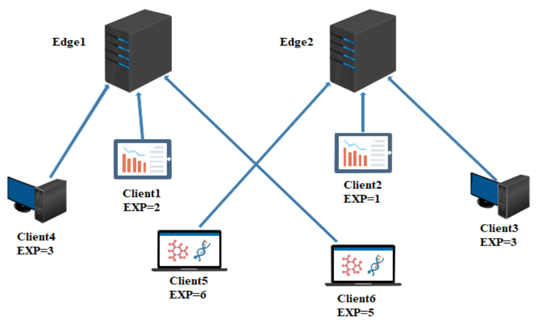


Fig. 3. Edge Server Diagram Connected to Different Clients.

For example, as depicted in Figure3, edge servers 1 and 2 are connected to different clients. For edge server 1, having already selected client1 and client4, if the selection were based solely on communication delay, the optimal choice would be client 5. However, our client selection algorithm takes into account both communication delay and the data heterogeneity of clients. Therefore, edge server 1 compares the EXP values of the two devices and, finding that EXP_6 is lower than EXP_5, selects client 6 to join the federated learning network. Similarly, for edge server 2, the COHEA algorithm was employed to account for client heterogeneity, leading to the selection of client 5.

Table 3. Algorithm 2 delineates the process of the COHEA algorithm.

```
Algorithm 2：Cost-Optimized Heterogeneous Client-Edge Association Algorithm
Input： the response latency $\{T_i\}_{i=1}^{N}$， Label distribution of all the clients $\{FD_k\}_{k=1}^{N}$
Output: client cluster for each edge node $c^j$ for $j \in \{1, 2,...,M\}$.
1: Initialize: client cluster for each edge node: $c^j = \varnothing$, $j \in \{1, 2, 3...,M\}$ ,and
         empty edge set for each client $s^k = \varnothing$, client $k \in \{1, 2, 3...,N\}$
2: Initialize: a client cluster $U = \{1, 2, 3...,N\}$
3:  while $U \neq \varnothing$ do
4:    for $j \in \{1, 2, 3...,M\}$ do
5:      if $c^j = \varnothing$ then
6:          Select the client k with the minimum response latency
7:          And set $c^j = \{k\}$
8:          Remove client k from U
9:      else
10:         Select client k by minimizing(19)
11:           $s^k = s^k \cup \{j\}$
12:     end if
13:   end for
14:   for $k \in \{1, 2, 3...,N\}$ do
15:     if $s^k \neq \varnothing$ then
16:         Randomly select an edge node j from $s^k$
17:           $c^j = c^j \cup \{k\}$
18:         Remove client k from U and clear $s^k$
19:     end if
20:   end for
21: end while
22: return $c^j$ for $j \in \{1, 2,...,M\}$
```

4 Experiment

4.1 Simulation Settings

We designed a simulation environment using Matlab R2019a, considering a large number of work nodes, varying client proportions, edge nodes, and the quantity of local aggregation. The area covered by the edge computing network in the simulation environment measures 5 km × 5 km. The background noise level in the simulated environment is set at -100 dB, the communication range of edge nodes is 350 meters, the CPU frequency of edge nodes is 1.5 GHz, and the transmission power of edge nodes is 2.2 W.

The hierarchical federated learning system comprises 100 clients, 10 edge servers, and 1 cloud server. Each edge server is assigned an equal number of clients, and each client possesses an equal amount of training data. In this system, we undertook a classic image classification task using the MNIST dataset and the CIFAR-10 dataset for training. For the MNIST dataset and the CIFAR-10 dataset, we employed a Convolutional Neural Network (CNN) with 21,840 trainable parameters. This CNN model is used for the task of handwritten digit classification, encompassing 10 different categories. To train this model, we utilized the Mini-batch Stochastic Gradient Descent (SGD) algorithm, with each batch containing 20 samples. The initial learning rate was set at 0.01, updated at an exponentially decaying rate of 0.995 per epoch.

All experiments were conducted on a Windows 11 operating system, powered by a 12th Gen Intel(R) Core(TM) i7-12700 CPU at 2100 MHz, featuring 12 cores and 20 logical processors. The system was equipped with 16GB of RAM, and we used Python 3.7 and PyTorch on PyCharm 2020 version for our experiments. In our proposed hierarchical federated learning system, to simulate the characteristics and distribution of data from different clients in the real world, we have taken into account the scenarios of non-identically distributed (Non-IID) data and physical device heterogeneity. Our Non-IID setting involves the allocation of samples from two categories to each client, with 10 clients on each edge server covering a total of 5 label categories. The datasets between edge servers exhibit non-identical distributions. To simulate the physical heterogeneity of devices, we have categorized clients based on three bandwidth levels—100Mbps, 150Mbps, and 200Mbps—and also three CPU utilization rates—50%, 60%, and 70%. This approach aims to more closely resemble the heterogeneous characteristics of different devices in the real world, providing a more comprehensive and practical data simulation for our research.

4.2 Experimental process

Under both IID (Independent and Identically Distributed) and Non-IID settings, we compared our HiQuFlexAsync scheme with other federated learning approaches. The following is an introduction to these comparative schemes:

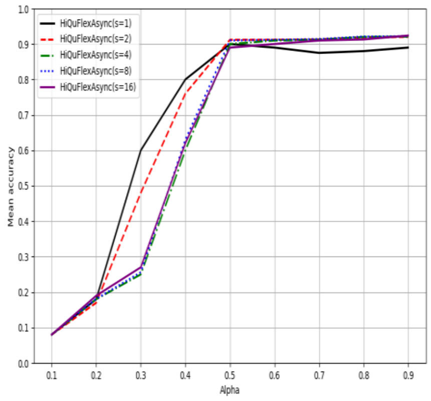
- FedAvg[13]: This is a cloud-based synchronous aggregation federated learning approach.
- FedAsync[14]: This is a cloud-based asynchronous federated learning algorithm. It updates the global model without waiting for slower clients to process.
- FedAT[15]: A hierarchical federated learning approach that combines synchronous intra-layer training with asynchronous inter-layer training. FedAT clusters clients based on their response delays, without considering the heterogeneity of client data.
- HierFAVG[16]: A hierarchical federated learning scheme based on cloud-edge-client architecture. It conducts synchronous updates in both client-edge aggregation and edge-cloud aggregation.
- HiFlash[17]: A layered federated learning paradigm that integrates mobile edge computing, employing synchronous client-edge model aggregation and asynchronous edge-cloud model aggregation.

During experimental phase, HiQuFlexAsync and its counterpart models consistently involve the same quantity of clients in every training round. First, we will explore the optimal value of α in HiQuFlexAsync, and then evaluate the performance of HiQuFlexAsync by comparing its test accuracy, loss data, and communication efficiency with those of other federated learning schemes.

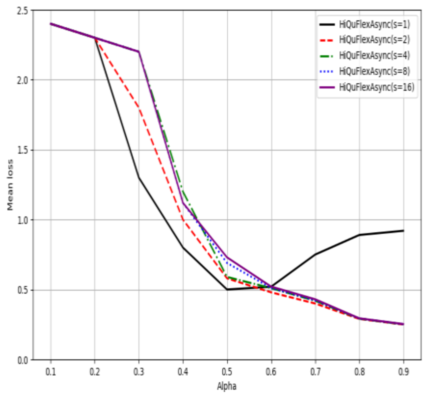
4. 3 Experimental Results and Analysis

A. Parameters Adjustment

Prior to comparing the performance with other federated learning methods, we conducted an assessment of the parameters in the proposed HiQuFlexAsync framework. The evaluation started with the asynchronous weight parameter α, with values systematically tested in the range of 0.1 to 0.9, aiming to determine the optimal α value. This process contributes to optimizing the performance of the HiQuFlexAsync model and provides robust support for subsequent performance comparisons.

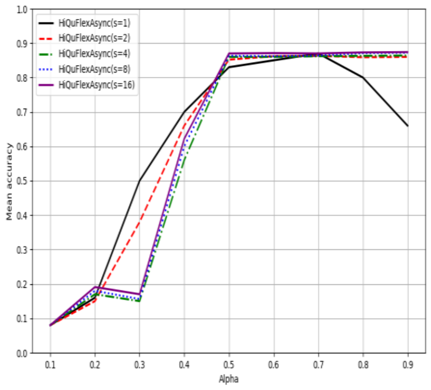


(a) Mean accuracy

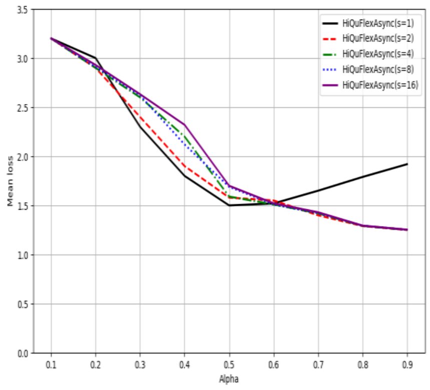


(b) Mean loss

Fig. 4. The mean values obtained by testing α from 0.1 to 0.9 on the MNIST dataset at quantization levels s of 1, 2, 4, 8, and 16, respectively.



(a) Mean accuracy

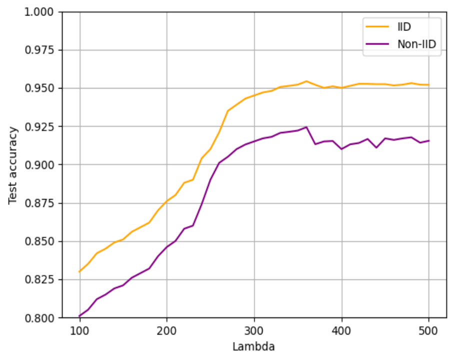


(b) Mean loss

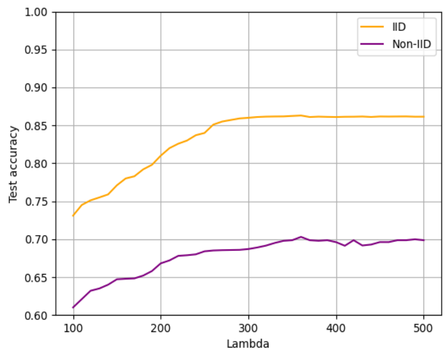
Fig. 5. The mean values obtained by testing α from 0.1 to 0.9 on the CIFAR-10 dataset at quantization levels s of 1, 2, 4, 8, and 16, respectively.

As shown in Figures 4 and 5 of the experimental results, during the gradual testing of α values from 0.1 to 0.9, it was observed that prior to setting α to 0.5, the model performance was suboptimal. Even after the same number of communication rounds, the loss function remained convergent at a higher level, and the model accuracy consistently lagged behind. However, upon setting α to 0.5, an improvement in model performance was noted. At this point, the average value of the loss function exhibited a decreasing trend, while the accuracy gradually increased. Further observations revealed that when α exceeded 0.6, the training performance of the model started to rebound, indicating a potential overfitting phenomenon. Therefore, the decision was made to confirm α = 0.5 as the optimal training value. Considering all factors, α was set to 0.5 for subsequent comparative experiments.

Subsequently, we evaluated the weight parameter λ by testing the model's accuracy and convergence time in HiQuFlexAsync under different λ values, aiming to select the optimal λ value.

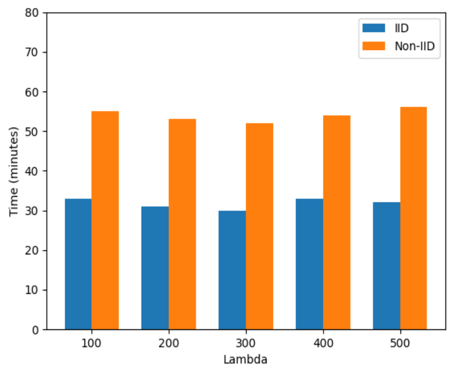


(a) MNIST

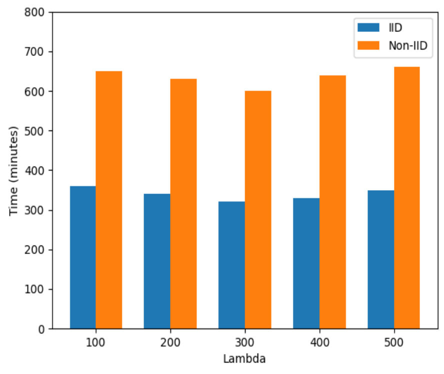


(b) CIFAR-10

Fig. 6. Fig.6 Test accuracy corresponding to different values of λ on the MNIST and CIFAR-10 datasets.



(a) MNIST



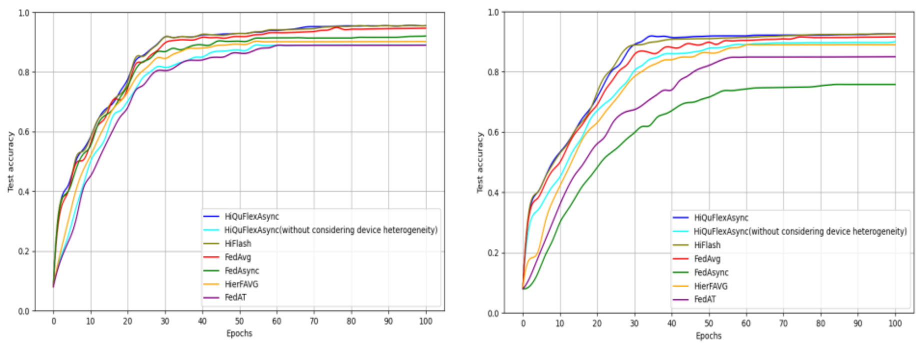
(b) CIFAR-10

Fig. 7. Convergence time corresponding to different values of λ on the MNIST and CIFAR-10 datasets.

From Figure 6, it can be observed that the test accuracy of HiQuFlexAsync reaches its peak when λ is between 350-400, regardless of whether in IID or Non-IID settings. Furthermore, Figure 7 indicates that the model's convergence time is minimal when λ falls within the range of 300-400. Taking into account both model accuracy and efficiency, λ value of 360 was chosen as the optimal parameter setting. In subsequent comparisons with other federated learning approaches, λ was fixed at 360.

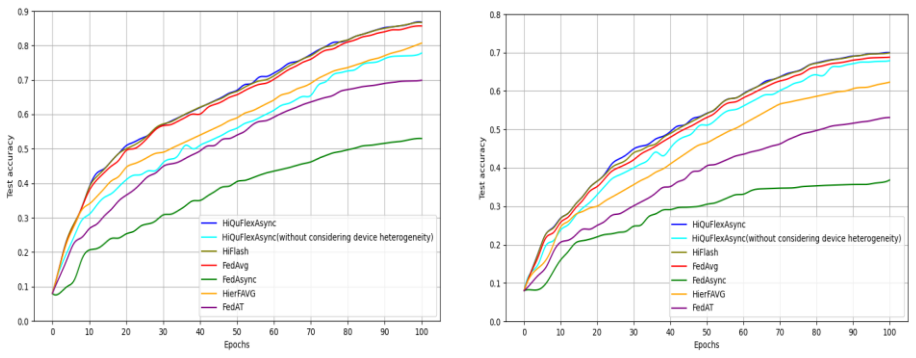
B. Model Accuracy Comparison

We conducted a performance comparison between HiQuFlexAsync and other federated learning schemes. After 100 training rounds on the MNIST dataset, the accuracy changes of various federated learning schemes are depicted in Figure 8. Similarly, after 100 training rounds on the CIFAR-10 dataset, the accuracy changes of various federated learning schemes are shown in Figure 9.



(a) IID (b) Non-IID

Fig. 8. Test accuracy of MNIST dataset under different data distributions w.r.t the total number of training epochs on the clients.



(a) IID (b) Non-IID

Fig. 9. Test accuracy of CIFAR-10 dataset under different data distributions w.r.t the total number of training epochs on the clients.

From Figures 8 and 9, it is evident that HiQuFlexAsync has achieved a level of training performance comparable to that of HiFlash. This can be attributed to our adoption of a three-tier federated learning architecture with quantization capability, integrating the COHEA algorithm. The key advantage of this algorithm lies in its consideration of client heterogeneity during the client allocation process, which promotes personalized optimization and efficient resource utilization, thereby enhancing model performance.

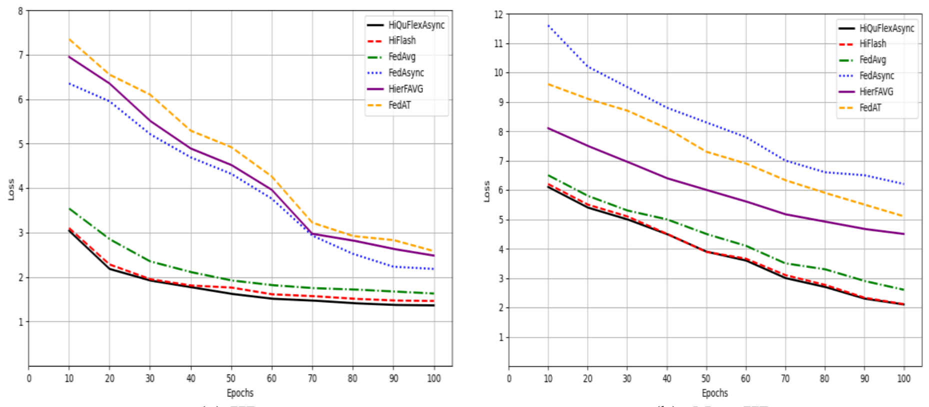
It is noteworthy that, in comparison to the three-tier architecture of HierFAVG and the asynchronous aggregation approach of FedAT, HiQuFlexAsync demonstrates a significant improvement in experimental accuracy. This improvement is attributed to our implementation of a hybrid architecture incorporating three tiers along with synchronous and asynchronous components, coupled with the utilization of the COHEA algorithm in the client allocation process.

Furthermore, we delved into the performance of the COHEA algorithm in HiQuFlexAsync without considering device heterogeneity. As illustrated in Figures 8 and 9, when disregarding device heterogeneity, the model accuracy of HiQuFlexAsync falls below the standard levels of conventional HiQuFlexAsync and HiFlash. This is attributed to the fact that accounting for device heterogeneity enables the full utilization of computing resources across different devices, thereby enhancing the overall computational performance and efficiency of the system. Without considering device heterogeneity, it results in a decrease in model accuracy. Hence, our algorithm takes device heterogeneity into account to enhance model performance.

C. Model Loss Comparison

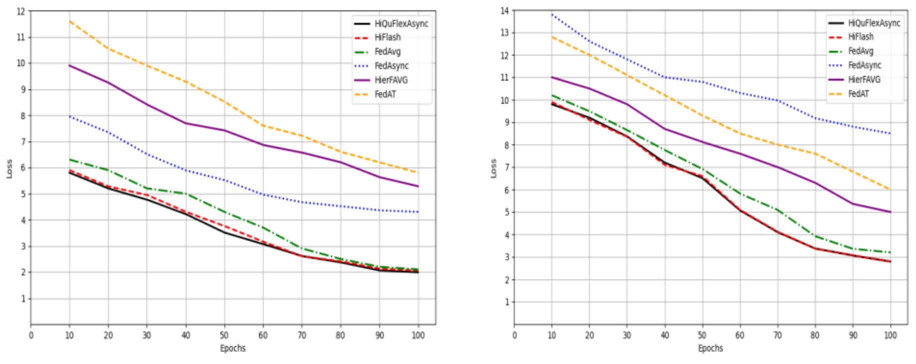
Figures 10 and 11 demonstrate the changes in loss values of various federated learning methods as the number of training rounds increases, under both IID and Non-IID settings for the MNIST and CIFAR-10 datasets.

Figures 10 and 11 depict the fitness and performance of various federated learning schemes during the training process. It is evident that HiQuFlexAsync has achieved significant fitting results in both IID and Non-IID settings, thanks to our more thoughtful consideration of client heterogeneity in the client allocation scheme. By optimizing client allocation to achieve load balancing and enhance system parallel processing capabilities, we have been able to elevate system performance. In comparison to alternative methods, our approach has demonstrated more pronounced enhancements in optimizing federated learning, further substantiating the effectiveness and feasibility of our methodology.



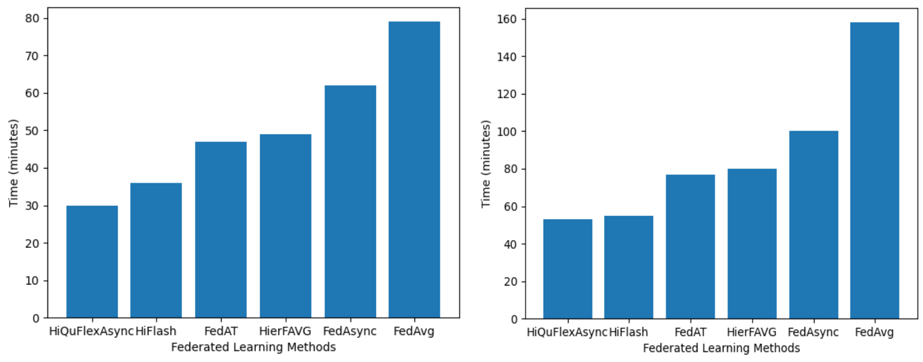
(a) IID (b) Non-IID

Fig. 10. loss values of MNIST dataset under different distributions w.r.t the total number of training epochs on the clients.



(a) IID (b) Non-IID

Fig. 11. loss values of CIFAR-10 dataset under different distributions w.r.t the total number of training epochs on the clients.



(a) IID (b) Non-IID

Fig. 12. Comparison of the time required by various federated learning methodologies to achieve 0.90 accuracy under the IID setting and 0.80 accuracy under the Non-IID setting on the MNIST dataset.

D. Model Efficiency Comparison

Further simulation experiments were conducted to investigate the time required for various federated learning schemes to achieve a specific level of accuracy on the MNIST dataset under IID and Non-IID settings. An accuracy target of 0.90 was chosen for the IID setting, while a target of 0.80 was selected for the Non-IID scenario. The selection of these values was based on prior experiments and a comprehensive consideration of the data complexity and comparability. The results of these experiments are presented in Figure 12.

Figure 12 vividly illustrates the high communication efficiency demonstrated by HiQuFlexAsync in both IID and Non-IID scenarios. This can be attributed to our model quantization, which has significantly bolstered communication efficiency. Furthermore, previous experimental findings suggest that while FedAVG excels in accuracy, it lags in communication efficiency compared to federated learning schemes employing a three-tier architecture and asynchronous updates. This deficiency arises from the enhanced parallelism and scalability offered by the three-tier architecture, coupled with the reduction in communication latency afforded by asynchronous updates. Conversely, the HiQuFlexAsync approach holistically considers both experimental accuracy and efficiency, thereby showcasing the exceptional performance of our methodology.

5 Conclusions

In the paper, a federated learning scheme named HiQuFlexAsync is proposed to address the challenges posed by data heterogeneity, physical heterogeneity, network latency, bandwidth limitations, and unstable network connections, which impact the performance of federated learning. Our approach adopts a cloud-edge-end three-tier architecture with quantization capability, improving communication efficiency through model quantization. Additionally, we introduce a client allocation algorithm called COHEA to optimize the client selection process, leveraging the data and physical heterogeneity of clients for federated learning. Furthermore, considering the communication constraints between cloud servers and edge servers, we adopt an asynchronous aggregation strategy to optimize the model aggregation process between them. Through extensive experimentation, we have demonstrated the superior performance and efficiency of the HiQuFlexAsync approach in training. However, we recognize that there is room for improvement in HiQuFlexAsync, such as exploring the impact of different network models on the algorithm. Therefore, future research will further explore and refine the HiQuFlexAsync scheme, extending its application to other domains.

Acknowledgments. This work was supposed by the Foundation of Chinese University Industry-University-Research Innovation (2022IT078), the Foundation of President of Hebei University (XZJJ202303).

6 Reference

1. Pan, Z., Sun, J., Li, X., Zhang, X., Bai, H.: Collaborative Face Privacy Protection Method Based on Adversarial Examples in Social Networks. In: Huang, DS., Premaratne, P., Jin, B., Qu, B., Jo, KH., Hussain, A. (eds) Advanced Intelligent Computing Technology and Applications. ICIC 2023. Lecture Notes in Computer Science, vol. 14086, pp. 499–510. Springer, Singapore (2023)
2. Chen, Y., Liang, L., Gao, W.: DFedSN: Decentralized federated learning based on heterogeneous data in social networks. World Wide Web , vol. 26, no. 5, pp. 2545–2568 (2023).
3. Mao, Y., You, C., Zhang, J., Huang, K., Letaief, K. B.: A survey on mobile edge computing:The communication perspective. IEEE Communications Surveys & Tutorials, vol. 19, no. 4, pp.2322-2358 (2017)
4. Nishio, T., Yonetani, R.: Client Selection for Federated Learning with Heterogeneous Resources in Mobile Edge. In: ICC 2019 - 2019 IEEE International Conference on Communications (ICC), pp. 1-7. Shanghai, China (2019)

5. Wang, N., Zhou, R., Su, L., Fang, G., Li, Z.: Adaptive Clustered Federated Learning for Clients with Time-Varying Interests. In: 2022 IEEE/ACM 30th International Symposium on Quality of Service (IWQoS), pp. 1-10. Oslo, Norway (2022)
6. Zheng, J., Li, K., Tovar, E., Guizani, M.: Federated Learning for Energy-balanced Client Selection in Mobile Edge Computing. In: 2021 International Wireless Communications and Mobile Computing (IWCMC), pp. 1942-1947. Harbin City, China (2021)
7. Chen, M., Yi, M., Huang, M., Huang, G., Ren, Y., Liu, A.: A novel deep policy gradient action quantization for trusted collaborative computation in intelligent vehicle networks. In: Processing (ICASSP), pp. 4118–4122 (2022)
8. Lu, Q., Murmann, B.: Enhancing the Energy Efficiency and Ro-bustness of TinyML Computer Vision Using Coarsely-Quantized Log-Gradient Input Images. ACM Transactions on Embedded Computing Systems, (2023)
9. Dupuy, C., Arava, R., Gupta, R., Rumshisky, A.: An Efficient DP-SGD Mechanism for Large Scale NLU Models. In: ICASSP 2022 -2022 IEEE International Conference on Acoustics, Speech and Signal (2022)
10. Luo, S., Chen, X., Wu, Q., Zhou, Z., Yu, S.: HFEL: Joint Edge Association and Resource Allocation for Cost-Efficient Hierarchical Federated Edge Learning. In IEEE Transactions on Wireless Communications, vol. 19, no. 10, pp. 6535-6548 (2020)
11. Lim, W. Y. B., et al.: Decentralized Edge Intelligence: A Dynamic Resource Allocation Framework for Hierarchical Federated Learning. In IEEE Transactions on Parallel and Distributed Systems, vol. 33, no. 3, pp. 536-550 (2022)
12. Chen, X., et al.: Two-Phase Deep Reinforcement Learning of Dynamic Resource Allocation and Client Selection for Hierarchical Federated Learning. In: 2022 IEEE/CIC International Conference on Communications in China (ICCC), pp. 518-523. Sanshui, Foshan, China (2022)
13. McMahan, H. B., Moore, E., Ramage, D., Hampson, S., Areas, B. A..: Communication-efficient learning of deep networks from decentralized data. In Proc. Int. Conf. Artif. Intell. Statist, pp. 1273–1282 (2017)
14. Xie, C., Koyejo, S., Gupta, I.: Asynchronous federated optimization. In Proc. NeurIPS Workshop Optim. Mach. Learn, pp. 1–11 (2020)
15. Chai, Z., Chen, Y., Anwar, A., Zhao, L., Cheng, Y., Rangwala, H.: FedAT: A High-Performance and Communication-Efficient Federated Learning System with Asynchronous Tiers. In: SC21: International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1-17. St. Louis, MO, USA (2021)
16. Liu, L., Zhang, J., Song, S. H., Letaief, K. B.: Client-Edge-Cloud Hierarchical Federated Learning. In: ICC 2020 - 2020 IEEE International Conference on Communications (ICC). pp. 1-6. Dublin, Ireland (2022)
17. Wu, Q., et al.: HiFlash: Communication-Efficient Hierarchical Federated Learning With Adaptive Staleness Control and Heterogeneity-Aware Client-Edge Association. IEEE Transactions on Parallel and Distributed Systems. pp. 1560–1579 (2023)