Multi-Width Neural Network-Assisted Hierarchical Federated Learning in Heterogeneous Cloud-Edge-Device Computing

Haizhou Wang
School of Computer Science and Technology, East China Normal University
Shanghai, China
51205901083@stu.ecnu.edu.cn

Guobing Zou
School of Computer Engineering and Science, Shanghai University
Shanghai, China
gbzou@shu.edu.cn

Fei Xu
School of Computer Science and Technology, East China Normal University
Shanghai, China
fxu@cs.ecnu.edu.cn

Yangguang Cui*
School of Computer Engineering and Science, Shanghai University
Shanghai, China
ygcui@shu.edu.cn

Tongquan Wei
School of Computer Science and Technology, East China Normal University
Shanghai, China
tqwei@cs.ecnu.edu.cn

Abstract

Federated learning (FL), an emerging data-secure distributed training paradigm, unites massive isolated Internet of Things (IoT) device nodes to collaboratively train a global neural network (NN) model without the exposure of their local multimedia data. However, constrained by the synchronous NN model integration nature of FL, there is a training latency inconsistency among heterogeneous devices, which significantly deteriorates FL training efficiency. Meanwhile, frequent local NN training and transmission impose high energy consumption pressure on users. To tackle these issues, this paper proposes a premium multi-width NN-assisted hierarchical FL (HFL) framework in heterogeneous cloud-edge-device computing to achieve remarkable training speedup and energy conservation. Specifically, a heterogeneity-aware NN width coefficient determination algorithm, which flexibly assigns a subnet with a suitable width to each user device based on its computing ability, is first applied to shorten the HFL training latency. Subsequently, to integrate subnets with different width topologies, we design a width-aware adaptive NN model integration approach to effectively ensure the accuracy of the integrated global NN model. Finally, a latency-aware energy saving strategy is introduced to reduce energy consumption. Experimental results demonstrate that our proposed framework outperforms state-of-the-art benchmarks, and attains up to 42.42% enhancement in accuracy, 81.5% reduction in training latency, and 40.9% optimization in energy cost.

*Corresponding author: Yangguang Cui

CCS Concepts

• **Computing methodologies → Distributed algorithms**.

Keywords

Hierarchical Federated Learning; Multi-Width Neural Network; System Heterogeneity; Training Latency; Energy Cost

ACM Reference Format:
Haizhou Wang, Guobing Zou, Fei Xu, Yangguang Cui, and Tongquan Wei. 2025. Multi-Width Neural Network-Assisted Hierarchical Federated Learning in Heterogeneous Cloud-Edge-Device Computing. In *Proceedings of the 33rd ACM International Conference on Multimedia (MM '25), October 27–31, 2025, Dublin, Ireland*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3746027.3754596

1 Introduction

Nowadays, artificial intelligence-empowered Internet of Things (IoT) applications provide remarkable intelligent services to humanity across various multimedia fields, such as autonomous driving, smart homes, and intelligent healthcare [1–4]. To offer high-quality intelligent services, it is crucial to collect sufficient training data for improving the performance of neural network (NN) models. However, in most real-world IoT scenarios, data generated from the device side is sensitive and can only be kept locally due to user privacy issues, which hinders a remote data center from collecting sufficient data to train a high-performance NN model [5–8].

To tackle the above obstacle, federated learning (FL), a forward-looking distributed paradigm, can organize collaborative distributed training for NN models among numerous devices while safeguarding user data privacy [9–11]. In a typical FL system, an edge server interacts with multiple devices over a wireless network, and the server-device cooperative training is executed iteratively [12–15]. In detail, a round of FL training proceeds as follows. Firstly, the edge server broadcasts the global NN model to devices. Secondly, each device trains the received global NN model using the local private dataset in parallel. Thirdly, each device uploads the trained NN model to the edge server in parallel. Lastly, the trained NN models, received by the edge server, are aggregated into a new global NN model via the *FedAvg* formula [16].

As information technology advances, the numerical scale of IoT devices has expanded dramatically [17, 18]. Unfortunately, the traditional edge server-coordinated FL system, built on a wireless network, fails to communicate with IoT devices located in distant places [19]. To deal with this challenge, a novel hierarchical FL (HFL) system, which relies on the cloud-edge-device architecture, is proposed [20, 21]. In HFL, a cloud server is linked to a few edge servers via the backbone network, and each edge server connects with some nearby user devices within a wireless network [19, 22]. Within this HFL system, each edge server is placed near user devices to reduce the transmission cost, and to enable the cloud server to offload transmission pressure onto edge servers.

In HFL, significant system heterogeneity exists in the computing ability among numerous IoT devices, and it takes more time for devices with weaker computing abilities to complete their local training of NN models [23]. Limited by the synchronization nature of FL, the slowest user device that completes local NN training and model transmission tasks with the longest latency, called the "straggler", determines the actual latency per distributed training round. In this context, even if some user devices with stronger computing capabilities can complete their current local training round of NN models quickly, they must still wait idly for synchronous NN model integration before commencing the next round. This idle waiting time hinders the entire HFL system from effectively leveraging the abundant computing resources available in certain heterogeneous devices [24, 25]. Consequently, the "barrel effect" caused by the resource-constrained "straggler" in a synchronous system significantly affect the HFL training efficiency.

Moreover, another issue that constrains the development of HFL systems is devices' high energy consumption cost [26]. Within HFL, devices are required to undertake the task of training local NN models on their local data and uploading the trained NN models to the edge servers. For the pursuit of portability, batteries are still used as a primary energy supply approach for IoT devices, which means that many user devices are constrained by limited energy [27]. As a result, the battery depletion from resource-intensive NN model training and transmission tasks may reduce computing speed and transmission stability, or even cause devices to shut down and disconnect, thus affecting the efficiency and reliability of HFL systems. In that context, it is urgent to explore an effective framework that is well suited for dealing with the above challenges.

This paper makes the following major contributions.

- We first propose a heterogeneity-aware NN width coefficient determination algorithm, which assigns customized subnets in the global multi-width NN model to devices with different computing abilities. In this way, heterogeneous devices can complete training tasks within a similar timeframe, thus achieving significant training acceleration.
- We then develop a width-aware adaptive NN model integration approach, which can accomplish adaptive NN model integration for subnets with different width topologies trained by user devices to effectively ensure NN model accuracy. In particular, the convergence analysis of our developed NN model integration scheme is rigorously proved.
- Finally, we introduce a latency-aware energy saving approach, which can align the time required by devices to complete each round of FL tasks. By adaptively lowering the operating frequency of devices that complete tasks before the unified deadline, this approach enables the FL training process to proceed in an energy-efficient manner.

2 System Models

2.1 Hierarchical Federated Learning

A generic HFL system, consisting of N heterogeneous devices $U = \{u_1, \cdots, u_N\}$, M edge servers $E = \{e_1, \cdots, e_M\}$ and a cloud server, is considered in this paper. Here, each device $u_n (1 \le n \le N)$ holds a local private dataset $D_n = \{x_{n,i}, y_{n,i}\}_{i=1}^{|D_n|}$ with a volume of $|D_n|$, where $x_{n,i}$ is the ith data instance and $y_{n,i}$ denotes its corresponding data label. Aided by the collaboration of the cloud server and edge servers, these N devices collaborate to execute HFL training tasks in parallel. Each user device u_n utilizes its dataset D_n to train its local neural network (NN) model W_n, that is [26]

$$min : \{L_n(W_n, D_n) = \frac{\sum_{i=1}^{|D_n|} l(W_n, x_{n,i}, y_{n,i})}{|D_n|}\}, \tag{1}$$

where $l(W_n, x_{n,i}, y_{n,i})$ is the loss function of the local NN model W_n at data point $\{x_{n,i}, y_{n,i}\}$.

Further, every edge server $e_m (1 \le m \le M)$ connects with a device set $U_m = \{u_1, \cdots, u_{|U_m|}\}$, and its dataset ϕ_m is expressed as $\phi_m = \cup_{u_n \in U_m} D_n$. Each edge server e_m and its device set U_m aim to explore a NN model $W_m^{'}$ that minimizes the loss function $L_m(W_m, \phi_m)$ on dataset ϕ_m, that is [28]

$$min : \{L_m(W_m^{'}, \phi_m) = \frac{\sum_{u_n \in U_m} \sum_{i=1}^{|D_n|} l(W_n, x_{n,i}, y_{n,i})}{|\phi_m|}\}. \tag{2}$$

In HFL, the global dataset Φ can be expressed as $\Phi = \cup_{1 \le m \le M} \phi_m$. The cloud server organizes all M edge servers and N user devices to perform HFL training, which aims at obtaining a global NN model W_G on the global dataset Φ to minimize the global loss function $L_G(W_G, \Phi)$, that is [19]

$$min : \{L_G(W_G, \Phi) = \frac{\sum_{m=1}^{M} L_m(W_m^{'}, \phi_m)}{|\Phi|}. \tag{3}$$

2.2 Multi-Width Neural Network

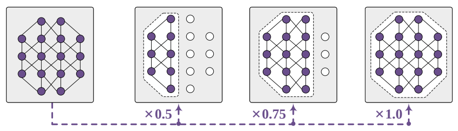


Figure 1: Illustration of multi-width neural network.

To accommodate heterogeneous devices with varying computational abilities, this paper introduces the multi-width neural network that can elastically switch to subnets $\{\omega_1, \omega_2, \cdots, \omega_K\}$ of varying widths, as shown in Fig. 1. Let K denote a hyper-parameter that defines the index number of subnets within the multi-width neural network. Within the multi-width neural network, a width list $\Upsilon = \{\varpi_1, \varpi_2, \cdots, \varpi_K\}$ is predefined. Here, a specific width coefficient $\varpi_k (1 \le k \le K)$ can be chosen to activate the corresponding proportion of neurons at each layer within the multi-width neural network, thereby switching to a subnet ω_k with width ϖ_k.

2.3 Local Calculation Cost Models

In HFL, devices adopt the widely used gradient descent to update model parameters. Given a learning rate α, the NN model W_n^p on the device u_n in the pth round is [29]

$$W_n^{p+1} = W_n^p - \alpha \nabla L_n(W_n^p, D_n), \tag{4}$$

where $\nabla L_n(W_n^p, D_n)$ denotes the calculated gradients of the loss function on local dataset D_n for the local model W_n^p.

In HFL, we use χ_n to represent the number of CPU cycles per second that the nth device u_n can execute, and the local training latency T_n^{cal} of device u_n can be expressed as [30]

$$T_n^{cal} = \frac{\sigma_n |D_n|}{\chi_n}, \tag{5}$$

where σ_n is the required CPU cycles for training NN models with a data point and $|D_n|$ is the total number of training data points on device u_n. As a result, $\sigma_n |D_n|$ CPU cycles are needed to deal with all data points on device u_n, and the training energy cost E_n^{cal} can be defined as [25]

$$E_n^{cal} = \varphi \sigma_n |D_n| {\chi_n}^2, \tag{6}$$

where φ denotes the effective switched capacitance concerning the chip architecture.

In this work, we plan to leverage the above-mentioned multi-width neural network in Subsection 2.2 to adapt to the devices' heterogeneity. Here, the multi-width neural network comprises K subnets $\{\omega_1, \cdots, \omega_k \cdots, \omega_K\}$ and each subnet has a distinct computational cost. This property allows the multi-width neural network to dynamically switch among subnets to accommodate heterogeneous devices with varying computational abilities and battery capacities. Hence, let σ_k denote the number of CPU cycles required to train one data point on the kth subnet ω_k of the multi-width neural network model on user device u_n, and Eq. (5) and Eq. (6) can be modified to Eq. (7) and Eq. (8), respectively, that is

$$T_n^{cal} = \frac{\sigma_k |D_n|}{\chi_n}, \qquad \sigma_k \in \{\sigma_1, \sigma_2, \cdots, \sigma_K\}, \tag{7}$$

$$E_n^{cal} = \varphi \sigma_k |D_n| {\chi_n}^2, \quad \sigma_k \in \{\sigma_1, \sigma_2, \cdots, \sigma_K\}. \tag{8}$$

2.4 Communication Cost Models

In HFL, each device u_n uploads the local NN model to its corresponding edge server e_m within one training round. This work considers a frequency-division multiple access (FDMA) sub-FL system which has B resource blocks (RBs) in total. For one user device u_n, let τ_n and υ_n denote the transmission power and the channel gain for device u_n, respectively. Thus, its data transmission rate R_n can be given as [19]

$$R_n = \vartheta_n B \log_2 (1 + \frac{\tau_n {\upsilon_n}^2}{N_0}), \tag{9}$$

where N_0 is the background noise, and $\vartheta_n (0 \leq \vartheta_n \leq 1.0)$ is the ratio of communication resources configured for each device u_n. As a result, the local transmission latency T_n^{com}, denoting the time slot for user device u_n to upload its selected subnet ω_k $(1 \leq k \leq K)$ to the server, is defined as [31]

$$T_n^{com} = \frac{\varsigma_k}{R_n}, 1 \leq k \leq K, \tag{10}$$

where ς_k symbolizes the data volume required to transmit subnet ω_k in bits. The transmission energy cost E_n^{com} for device u_n to transmit its subnet ω_k to the edge server is [32]

$$E_n^{com} = \tau_n T_n^{com}. \tag{11}$$

2.5 Latency and Energy Models within One Training Round

Let P denote the total number of HFL training rounds. Based on the synchronous fashion, the total latency $T_{m,p}^{edge}$ of pth training round of one sub-FL system coordinated by an edge server e_m is decided by the slowest one in set U_m, that is [31]

$$T_{m,p}^{edge} = \max_{u_n \in U_m} \{T_{n,p}^{total}\}, \tag{12}$$

where $T_{n,p}^{total}$ is the total latency of pth training round for each device u_n in set U_m. $T_{n,p}^{total}$ denotes the sum of the local training latency $T_{n,p}^{cal}$ and transmission latency $T_{n,p}^{com}$ of device u_n within pth training round, and $T_{n,p}^{total} = T_{n,p}^{cal} + T_{n,p}^{com}$.

Similarly, the global latency T_p^{global} for the overall HFL system in pth training round depends on the slowest edge server-coordinated sub-FL system, that is [19]

$$T_p^{global} = \max\{T_{1,p}^{edge}, T_{2,p}^{edge}, \cdots, T_{M,p}^{edge}\}. \tag{13}$$

With respect to energy cost, in one sub-FL system coordinated by edge server e_m, the total energy cost $E_{m,p}^{edge}$ of pth training round covering its device set U_m is defined as [26]

$$E_{m,p}^{edge} = \sum_{u_n \in U_m} \{E_{n,p}^{cal} + E_{n,p}^{com}\}, \tag{14}$$

where $E_{n,p}^{cal}$ and $E_{n,p}^{com}$ are the training energy cost and transmission energy cost of the device $u_n (u_n \in U_m)$ within pth round, respectively. Similarly, the global energy cost E_p^{global} of pth training round in this HFL system is defined as [19]

$$E_p^{global} = \sum_{m=1}^{M} E_{m,p}^{edge}. \tag{15}$$

3 Our Proposed Framework

In this work, we design a multi-width neural network-assisted efficient hierarchical federated learning (HFL) framework, which is detailed in **Algorithm 1**. The workflow of this algorithm can be summarized into the following two parts: the initialization part in lines 1-6 and the iterative training part in lines 7-21.

In the initialization part, for each edge server-coordinated sub-FL system, each edge server e_m first broadcasts requests to all devices in its device set U_m for collecting device-related resource information in line 2; Then, the information of all devices in U_m is transmitted to the cloud server via edge server e_m in line 3. Based on the device-related resource information, the cloud server adopts **Algorithm 2** to choose optimal subnets from the subnet sampling pool of the global multi-width NN model for these devices in line 5, and call **Algorithm 4** to decide the device operating frequency list for these devices in line 6.

Within the second part, P rounds of HFL training are iteratively performed. At the beginning of each round, the latest global multi-width NN model held by the cloud server is first delivered to all edge servers in line 8. Then, all edge servers parallelly execute NN model training and transmission tasks within their respective sub-FL systems in lines 9-12. Specifically, an edge server e_m first organizes its devices to perform local FL training in parallel in line 10, which is detailed in **Procedure LOCAL_TRAIN**. Then, the edge server e_m conveys trained subnet models uploaded by its corresponding devices to the cloud server in line 11. After trained subnet models from all M edge servers are received, the cloud server adopts **Algorithm 3** to integrate different subnet models into a new global multi-width NN model W_G^{p+1} in line 13. Lastly, if the target training round has not arrived, the algorithm continues the next round from line 7.

In **Procedure LOCAL_TRAIN**, lines 16-21 detail the local training process within one sub-FL system coordinated by an edge server. In line 18, each edge server delivers a customized subnet into each device in its device set, and this customized subnet is determined by the cloud server via **Algorithm 2**. Due to the adaptation of subnets to heterogeneous devices with various computational abilities, these customized devices can complete training in parallel around the same timeline, thereby avoiding devices with weaker computational abilities to slow down the entire training process. Then, each user device u_n with its customized subnet ω_{π_n} utilizes the local dataset D_n to perform local training using Eq. (4) in line 19. Finally, the subnet $\omega_{\pi_n,n}$ trained by device u_n is transferred to its edge server u_m in line 20.

4 User Heterogeneity-Aware NN Width Coefficient Determination

4.1 Motivation and Rationality Analysis

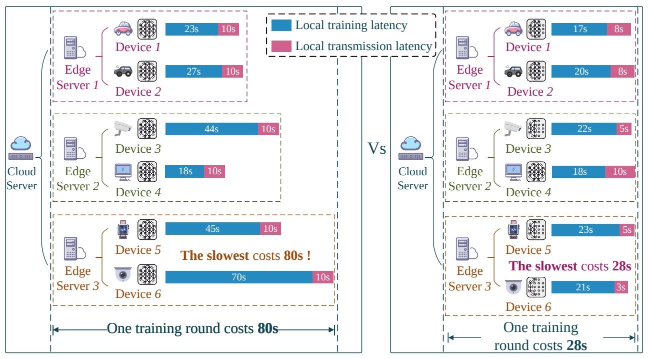
In HFL, the structure of the NN model largely determines the local calculation and communication overhead for devices. This fact inspires us to design a multi-width NN-assisted efficient HFL framework. To visualize the impact of NN model structure on HFL training, we first construct an HFL system consisting of 1 cloud server, 3 edge servers, and 6 heterogeneous devices, where each edge server collaborates with 2 devices. Subsequently, we configure the NN model structure as either the traditional NN model or the multi-width NN model to compare the training time cost, as shown in Fig. 2. Specifically, when using the conventional NN model structure, Fig. 2(a) exhibits the time cost of one training round in HFL. It is seen that the 6th device, with the worst computational capability among all heterogeneous devices, spends 70 seconds completing local training of the NN model and 10 seconds uploading NN model. Limited to the synchronous mechanism of FL, devices with stronger computing capabilities remain idle until the slowest device completes its FL task, which significantly reduces training efficiency.

Fig. 2(b) shows the time cost of one training round in HFL based on the multi-width NN model. By assigning narrow subnets to devices with poor computational abilities, the local training time and model upload latency of these devices are drastically reduced. As a result, the overall training time cost for one HFL training round is drastically reduced from 80 seconds to 28 seconds by using the multi-width NN model.

Algorithm 1: Our proposed framework

```
   Input: A cloud server, $M$ edge servers $\{e_1, \cdots, e_M\}$ with $M$ device sets
          $\{U_1, \cdots, U_M\}$, the maximum training round $P$, the initial global
          multi-width NN model $W_G$;
   Output: The well-trained global multi-width NN model $W_G^P$;
1  for $m = 1$ to $M$ parallelly do
2      Edge server $e_m$ distributes requests to its user device set $U_m$ to collect
         device-related resource information;
3      All user devices in $U_m$ upload their device-related resource information
         to the cloud server through the edge server $e_m$;
4  end
5  According to the device-related resource information collected, call
     Algorithm 2 to assign a subnet index $\pi_n$ $(1 \le \pi_n \le K)$ for each user
     device $u_n$ $(1 \le n \le N)$ through $M$ edge servers;
6  Based on the assigned subnet information, call Algorithm 4 to decide the
     device operating frequency $\chi_n^*$ of each user device $u_n$;
7  for $p = 0$ to $(P - 1)$ do
8      The cloud server delivers the latest global multi-width NN model $W_G^p$ to
         $M$ edge servers;
9      for $m = 1$ to $M$ parallelly do
10         Edge server $e_m$ calls Procedure LOCAL_TRAIN to organize
             devices in its device set $U_m$ to perform local training;
11         Edge server $e_m$ transfers trained subnet models received from its
             user devices in $U_m$ to the cloud server;
12     end
13     The cloud server calls Algorithm 3 to merge multiple subnet models as
         the latest global multi-width NN model $W_G^{p+1}$;
14 end
15 Return The well-trained global multi-width NN model $W_G^P$ ;

16   Procedure LOCAL_TRAIN ($e_m$, $U_m$)
17 for Each user device $u_n$ in the device set $U_m$ parallelly do
18     Edge server $e_m$ delivers a customized subnet $\omega_{\pi_n}$ to user device $u_n$ in
         the device set $U_m$;
19     Each user device $u_n$ trains subnet $\omega_{\pi_n}$ with local dataset $D_n$ by using
         Eq. (4);
20     Each user device $u_n$ transfers the trained subnet $\omega_{\pi_n,n}$ to its
         coordinating edge server $e_m$;
21 end
```



(a) The traditional neural network (b) The multi-width neural network

Figure 2: Time cost comparison based on the conventional NN and multi-width NN.

4.2 User Heterogeneity-Aware NN Width Coefficient Determination Algorithm

As revealed in the previous subsection 4.1, considering that the unignorable divergences of computational ability exist among heterogeneous devices in HFL [23, 25], we introduce a user heterogeneity-aware NN width coefficient determination algorithm that allocates subnets with appropriate widths for various heterogeneous devices, as explained in **Algorithm 2**. This algorithm can be summarized into three parts: the initialization part in lines 1-2, the benchmarking timeline determination part in lines 3-9, and the subnet index assignment part in lines 10-22.

Within the first part, according to the predefined subnet width list, a multi-width NN model containing K subnets is built in line 1. Then, the benchmarking timeline $T_{benchmark}$, denoting the expected timeline needed for devices to complete training, is initialized to infinity in line 2. Within the second part, in lines 3–9, the minimal delay across all N devices using the full-width subnet is set as the benchmarking timeline. Each device u_n switches to subnet ω_K in line 4, then obtains its local training and transmission latencies in lines 5–6. Their sum, T_n^{total}, is calculated in line 7, and the benchmark timeline is updated to the smaller value between T_n^{total} and itself in line 8. This process repeats over all devices to finalize $T_{benchmark}$.

The last part iteratively assigns suitable subnet indexes for devices in lines 10-22. Firstly, the timeline gap T_n^{gap}, which represents the difference between $T_{benchmark}$ and the device's latency T_n^{total}, is initialized to infinity in line 11. Then, u_n searches subnet widths from wide to narrow in lines 12-21. Device u_n switches to subnet

Algorithm 2: NN width coefficient determination

```
Input: The predefined subnet width list $\Upsilon = \{\varpi_1, \cdots, \varpi_K\}$;
Output: A subnet width index list $\{\pi_1, \pi_2, \cdots, \pi_N\}$ for devices;
1  Construct a multi-width NN model $W$ with $K$ subnets $\{\omega_1, \cdots, \omega_K\}$ based
     on the subnet width list $\{\varpi_1, \cdots, \varpi_K\}$;
2  Initialize the benchmarking timeline $T_{benchmark}$ as $\infty$;
3  for $n = 1$ to $N$ do
4      Each device $u_n$ switches to the full-width subnet $\omega_K$ in multi-width NN
         model $W$;
5      Calculate the local training latency $T_n^{cal}$ of device $u_n$ with the
         full-width subnet $\omega_K$ by using Eq. (7);
6      Calculate the local transmission latency $T_n^{com}$ of device $u_n$ with the
         full-width subnet $\omega_K$ by using Eq. (10);
7      Obtain the sum of local training latency and local transmission latency
         $T_n^{total} = T_n^{cal} + T_n^{com}$;
8      Refresh the benchmarking timeline $T_{benchmark}$ as
         $T_{benchmark} = \min\{T_{benchmark}, T_n^{total}\}$;
9  end
10 for $n = 1$ to $N$ do
11     Initialize the timeline gap $T_n^{gap} = \infty$ for the user device $u_n$;
12     for $k = K$ to 1 do
13         Each device $u_n$ switches to the subnet $\omega_k$ with the width
             coefficient $\varpi_k$ in the multi-width NN model $W$;
14         Calculate the local training latency $T_n^{cal}$ of device $u_n$ with subnet
             $\omega_k$ by using Eq. (7);
15         Calculate the local transmission latency $T_n^{com}$ of device $u_n$ with
             subnet $\omega_k$ by using Eq. (10);
16         Record the sum of local NN model training latency and
             transmission latency $T_n^{total} = T_n^{cal} + T_n^{com}$;
17         if $|T_{benchmark} - T_n^{total}| < T_n^{gap}$ then
18             Update the subnet width index $\pi_n$ of device $u_n$ to $k$;
19             Refresh the timeline gap $T_n^{gap}$ of device $u_n$ as
                 $T_n^{gap} = |T_{benchmark} - T_n^{total}|$;
20         end
21     end
22 end
23 Return Subnet width index list $\{\pi_1, \cdots, \pi_N\}$ for user devices;
```

ω_k with corresponding width coefficient ϖ_k in line 13, and computes its total latency T_n^{total} in lines 14-16 Afterwards, in line 17, if the difference between $T_{benchmark}$ and T_n^{total} for the current subnet is less than the timeline gap T_n^{gap}, the subnet index π_n of device u_n is updated to k and the timeline gap T_n^{gap} is updated to $|T_{benchmark} - T_n^{total}|$ in lines 18-19.

5 Width-Aware NN Model Integration

5.1 Design Details

As described in Section 4, devices are assigned subnets with unequal widths for accommodating their different computing capabilities. Here, after devices have completed the training updates of subnets, the traditional *FedAvg* formula cannot be directly applied to NN model integration due to the different NN model structures.

Fig. 3 shows a visual illustration for our designed width-aware adaptive NN model integration scheme. After the subnet models trained by devices are transferred to the cloud server via multiple edge servers, our scheme is used to merge these multi-width subnets with different topological structures. Specifically, the cloud server first splits each subnet model into corresponding subnet areas. Then, for subnet areas in the same position, we weightedly aggregate the corresponding parameters to build a new subnet area. Subsequently, these updated subnet areas in different positions are connected, and yield a new global multi-width NN model that wisely incorporates knowledge from all subnet models. Here, considering the pth training round in our HFL system, each device u_n utilizes its local dataset D_n to train a subnet $\omega_{k,n}$ with width coefficient k, and then u_n uploads the well-trained subnet $\omega_{k,n}$ to the cloud server via its corresponding edge server. Formally, our width-aware adaptive NN model integration scheme can be formulated as follows

$$W^{p+1} = \frac{\sum_{u_n \in \mathfrak{U}_K} |D_n| (\omega_{K,n}^{p+1} - \omega_{K-1,n}^{p+1})}{|\mathcal{D}_{\mathfrak{U}_K}|} + \cdots + \frac{\sum_{u_n \in \mathfrak{U}_2} |D_n| (\omega_{2,n}^{p+1} - \omega_{1,n}^{p+1})}{|\mathcal{D}_{\mathfrak{U}_2}|} + \frac{\sum_{u_n \in \mathfrak{U}_1} |D_n| \omega_{1,n}^{p+1}}{|\mathcal{D}_{\mathfrak{U}_1}|}; \tag{16}$$

where $\mathfrak{U}_k = \bigcup_{\pi_n \geq k} u_n$ denotes the union of user devices that contains the kth subnet area, and $|\mathcal{D}_{\mathfrak{U}_k}| = \sum_{u_n \in \mathfrak{U}_k} |D_n|$ is the total dataset volume of user devices in $\mathfrak{U}_k$.

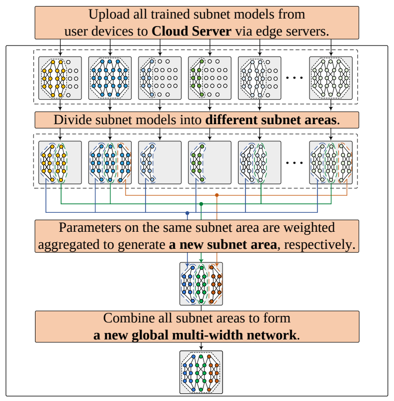

Upload all trained subnet models from user devices to Cloud Server via edge servers.
Divide subnet models into different subnet areas.
Parameters on the same subnet area are weighted aggregated to generate a new subnet area, respectively.
Combine all subnet areas to form a new global multi-width network.

Figure 3: Illustration for our designed width-aware adaptive NN model integration scheme.

Algorithm 3: Width-aware adaptive NN model integration

```
Input: A global multi-width NN model $W^p$ in $p$th training round, list of
    subnets from $N$ devices $\{\omega_{\pi_1,1},\cdots,\omega_{\pi_N,N}\}$;
Output: The latest global multi-width NN model $W^{p+1}$;
1  for $k = 1$ to $K$ do
2      Initialize the device set that contains the $k$th subnet area as $\mathfrak{U}_k = \varnothing$;
3      Initialize the dataset volume of device set $\mathfrak{U}_k$ as $|\mathcal{D}_{\mathfrak{U}_k}| = 0$;
4      for $n = 1$ to $N$ do
5          if $\pi_n \geq k$ then
6              Update user device set $\mathfrak{U}_k = \mathfrak{U}_k \cup u_n$;
7              Refresh the dataset volume $|\mathcal{D}_{\mathfrak{U}_k}| = |\mathcal{D}_{\mathfrak{U}_k}| + |D_n|$;
8          end
9      end
10 end
11 Cloud server integrates all $N$ subnets with different widths into a new global
       multi-width NN model $W^{p+1}$ by using Eq. (16);
12 Return The merged global multi-width NN model $W^{p+1}$;
```

The above workflow of our width-aware adaptive NN model integration approach is detailed in **Algorithm 3**. Lines 1-10 sequentially collect the information related to K subnet areas. Specifically, let $\mathfrak{U}_k$ denote the device set that contains the kth subnet area, and it is first constructed as an empty set in line 2. In line 3, let $|\mathcal{D}_{\mathfrak{U}_k}|$ denote the total dataset volume of devices in $\mathfrak{U}_k$, and $|\mathcal{D}_{\mathfrak{U}_k}|$ is initialized as zero. Subsequently, we iteratively mark the device set $\mathfrak{U}_k$ used for the kth subnet area integration and the data volume information in lines 4-9. After collecting all subnet area information, according to Eq. (16), the cloud performs width-aware adaptive NN model integration for the subnets with different widths in line 11. Finally, line 12 returns the merged global multi-width NN model.

5.2 Convergence Analysis

Here, we analyze the training convergence of our proposed framework. The primary distinctions of our proposed solution lie in the different width subnet training on devices and width-aware adaptive NN model integration. Firstly, in the pth training round, the training update of the adopted subnet ω_k with width coefficient π_k on device u_n is

$$\omega_{k,n}^{p+1} = \omega_{k,n}^{p} - \alpha_p \nabla L_n^p(\omega_{k,n}^p, \wp_n^p), \tag{17}$$

where $\wp_n^p$ is the stochastic gradient sample. Secondly, the width-aware NN model integration scheme on cloud server is

$$W^{p+1} = \frac{\sum_{u_n \in \mathfrak{U}_K} |D_n|(\omega_{K,n}^{p+1} - \omega_{K-1,n}^{p+1})}{|\mathcal{D}_{\mathfrak{U}_k}|} + \cdots + \frac{\sum_{u_n \in \mathfrak{U}_2} |D_n|(\omega_{2,n}^{p+1} - \omega_{1,n}^{p+1})}{|\mathcal{D}_{\mathfrak{U}_2}|} + \frac{\sum_{u_n \in \mathfrak{U}_1} |D_n|\omega_{1,n}^{p+1}}{|\mathcal{D}_{\mathfrak{U}_1}|}; \tag{18}$$

Taking Eq. (17) into Eq. (16), we can obtain

$$W^{p+1} = W^p - \underbrace{\alpha_p \sum_{k=K}^{1} \frac{\sum_{u_n \in \mathfrak{U}_k} (\nabla L_n^p(\omega_{k,n}^p, \wp_n^p) - \nabla L_n^p(\omega_{k-1,n}^p, \wp_n^p))}{|\mathcal{D}_{\mathfrak{U}_k}|}}_{\Theta^p};$$

where Θ^p represents the gradient update information in this round, and let $\bar{\Theta}^p$ denote the gradient update with the full sampling method. Here, some well-recognized assumptions in literatures [17, 33, 34] are first presented as follows.

Assumption 1: *For all* $\mathbf{a}$ *and* $\mathbf{b}$, $\{L_1,\cdots,L_N\}$ *are L-smooth, that is*

$$L_n(\mathbf{a}) \leq L_n(\mathbf{b}) + (\mathbf{a}-\mathbf{b})^T \nabla L_n(\mathbf{b}) + \frac{L}{2}||\mathbf{a}-\mathbf{b}||_2^2.$$

Assumption 2: *For all* $\mathbf{a}$ *and* $\mathbf{b}$, $\{L_1,\cdots,L_N\}$ *are* μ*-strong convex, that is*

$$L_n(\mathbf{a}) \geq L_n(\mathbf{b}) + (\mathbf{a}-\mathbf{b})^T \nabla L_n(\mathbf{b}) + \frac{\mu}{2}||\mathbf{a}-\mathbf{b}||^2.$$

Assumption 3: *The local training variance of stochastic gradient updates on each device* u_n *is upper bounded, that is*

$$E||\nabla L_n^p(W_n^p, \wp_n^p) - \nabla L_n^p(W_n^p)||^2 \leq \sigma^2.$$

Before carrying out the training convergence analysis, according to Eq. (16) and Assumptions 1, 2, and 3, we can prove the following Lemmas 1 and 2.

Lemma 1: With the assumption 3, the variance Θ^p of the global training gradient is bounded as follows

$$E||\Theta^p - \bar{\Theta}^p||^2 \leq \Upsilon,$$

where $\Upsilon = \sum_{k=K}^{1} \frac{\sum_{u_n \in \mathfrak{U}_k} \sigma^2}{|\mathcal{D}_{\mathfrak{U}_k}|}$.

Lemma 2: With the learning rate $\alpha_p \leq \frac{1}{L}$ and assumptions 1 and 2, the discrepancy between the global model update and the optimal training process in each round satisfies

$$E||W^{p+1} - W^*||^2 \leq (1-\mu\alpha_p)E||W^p - W^*||^2 + \alpha_p^2\Upsilon.$$

With the Lemmas 1 and 2, we can complete the training convergence analysis, as illustrated by the **Theorem**.

Theorem: With the assumptions 1, 2, and 3, training convergence guarantee for our proposed framework is

$$E[L(W^p)] - L(W^*) \leq \frac{\mu L^2 \Delta_1 + 2\Upsilon L}{\mu^2 p + 2L\mu - \mu^2}.$$

where $\Upsilon = \sum_{k=K}^{1} \frac{\sum_{u_n \in \mathfrak{U}_k} \sigma^2}{|\mathcal{D}_{\mathfrak{U}_k}|}$ and $\alpha_p = \frac{2}{\mu p + 2L - \mu}$.

Obviously, with the training rounds $p \rightarrow \infty$, $(E[L(W^p)] - L(W^*)) \rightarrow 0$. Thus, its training convergence is guaranteed.

6 Latency-Aware Energy Saving Strategy

Due to the complexity of NN design considerations, the number of subnet widths is generally given. Under a given subnet width list, the most suitable subnet width indexes are assigned to heterogeneous devices. In this context, when certain devices switch to the customized subnets, they may complete the local training task slightly faster than the maximum training latency. Thus, there is idle time that cannot be effectively utilized due to the synchronous mechanism. Oftentimes, almost all user devices support dynamic voltage/frequency scaling (DVFS) techniques in reality, and each device u_n operates at its highest CPU frequency χ_n^{max} by default. Inspired by this, this work aims to explore an energy saving strategy, as elaborated in **Algorithm 4**.

This algorithm first initializes the maximum training latency T_{max} within the current training round to 0 in line 1. Then, lines 2-7 sequentially compute the local NN model training and transmission latency for all devices, and continuously renew the maximum training latency T_{max}. Specifically, equipped with the corresponding subnet ω_{π_n}, the local training latency T_n^{cal} and transmission latency T_n^{com} for each device u_n can be calculated in lines 3-4, and these are summed up to obtain its overall training delay T_n^{total}

Algorithm 4: DVFS-based training energy saving

```
Input: A subnet width index list $\{\pi_1, \pi_2, \cdots, \pi_N\}$;
Output: The device operating frequency list $\{\chi_1^*, \chi_2^*, \cdots, \chi_N^*\}$;
1  Initialize the maximum training latency $T_{max}$ in this round as 0;
2  for $n = 1$ to $N$ do
3      Calculate the local training latency $T_n^{cal}$ of device $u_n$ with its assigned
         subnet $\omega_{\pi_n}$ by using Eq. (5);
4      Calculate the local transmission latency $T_n^{com}$ of device $u_n$ with its
         assigned subnet $\omega_{\pi_n}$ by using Eq. (10);
5      Obtain the sum of local training latency and local transmission latency
         $T_n^{total} = T_n^{cal} + T_n^{com}$;
6      Renew the the maximum training latency $T_{max}$ within the current
         training round as $T_{max} = \max\{T_{max}, T_n^{total}\}$;
7  end
8  for $n = 1$ to $N$ do
9      Modify the device operating frequency of user device $u_n$ to
         $\chi_n^* = \max(\frac{\sigma_k |D_n|}{T_{max} - T_n^{com}}, \chi_n^{min})$;
10 end
11 Return The device operating frequency list $\{\chi_1^*, \chi_2^*, \cdots, \chi_N^*\}$;
```

used for completing one FL training round on the device u_n in line 5. In line 6, according to the overall training delay T_n^{total} of the current user device, the maximum training delay T_{max} is updated as $T_{max} = \max\{T_{max}, T_n^{total}\}$. Subsequently, lines 8-10 sequentially tune the users' device operating frequencies. Finally, a tuned device operating frequency list for N devices is obtained. Based on the idle time, this strategy can effectively reduce energy cost.

7 Evaluation

7.1 Experimental Settings

A generic hierarchical FL system, containing one cloud server and five edge server-coordinated sub-FL systems, is considered in this paper. Within each edge server-coordinated sub-FL system, an edge server connects with five heterogeneous devices. Here, the minimum CPU operating frequency for each device is deemed as 0.3 GHz [19], while the maximum CPU operating frequency is sampled from the range of [0.3, 2.0] GHz [19]. As in [35], it is set that all devices can perform one MAC operation per CPU cycle. The effective switched capacitance is assigned a value of 2×10^{-28} [30]. In terms of communication parameters, the overall resource blocks B are given as 2 MHz [32] and the background noise N_0 is 10^{-9} W [36]. The ratio of communication resources ϑ and transmission power τ for devices are assumed to be 0.2 and 0.2 W [30].

This work utilizes popular CIFAR-10 [19], CIFAR-100 [31] and SVHN [37] datasets to carry out performance comparison, and the training data are randomly shuffled and dispensed to devices. In HFL training, the well-known Wide Residual Networks [38] (WRN) model is used for CIFAR-10 and CIFAR-100 datasets and Multi-layer Perceptron (MLP) [35] model is employed for SVHN dataset. Following the design principles of our solution, this WRN model is tuned by the width list [0.35, 0.65, 1.0] to a multi-width WRN model, named WRN-[0.35, 0.65, 1.0], which consists of three subnets with widths of 0.35, 0.65, and 1.0, respectively; let MLP-[0.35, 0.65, 1.0] denote a multi-width MLP model with a width list [0.35, 0.65, 1.0], and it has three MLP subnet models with widths of 0.35, 0.65, and 1.0, respectively. To substantiate the effectiveness of our proposed method, we compare it with the well-known benchmarks. *Original FL* [16] refers to the original FL. *FedCS* [39] prioritizes devices with stronger computing resources to complete FL training within the specified deadline. *HFL* [17] denotes a generic hierarchical FL. *HFedCS* expands *FedCS* to the hierarchical FL system. *SL* [40] is a separate training scheme without NN model aggregation.

7.2 Results on Training Accuracy

Figs. 4(a)-4(c) illustrate the accuracy curve of our method compared to these of 5 benchmarks on CIFAR-10, CIFAR-100, and SVHN datasets. It can be seen that our scheme exhibits a better accuracy curve. For instance, on the CIFAR-10 dataset, as shown in Fig. 4(a), our method achieves the best training result with 89.21% accuracy, surpassing 58.51%, 81.48%, 70.02%, 85.96%, and 88.96% accuracy of SL, ClassicFL, FedCS, HFedCS, and HFL benchmarks, respectively. Similarly, in the CIFAR-100 and SVHN datasets, consistent experimental observations can be obtained, as shown in Figs. 4(b) and 4(c). Overall, compared to benchmarks, our scheme can achieve up to 42.42% accuracy benefits.

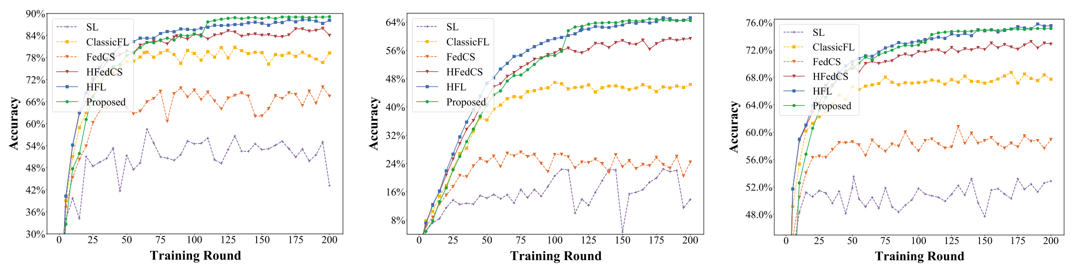
Here, we elaborate on the reasons for accuracy performance results. The lowest accuracy of SL is attributed to its separate training mode. In addition, ClassicFL and FedCS fail to take full advantage of the fact that the cloud server can accommodate more devices and training data. For HFedCS and HFL, HFedCS excludes slower devices from the training process to speed up training, which reduces data diversity; our method and HFL achieves similar accuracy. Meanwhile, by deploying subnets of appropriate widths on heterogeneous devices, our method has significant advantages in terms of training acceleration and energy cost.

7.3 Results on Training Latency

Table 1 presents the training latency of all methods, including our method and benchmarks, to achieve various expected accuracies on the CIFAR-10, CIFAR-100, and SVHN datasets. As observed, our method can obtain significant latency optimization. For instance, results on the CIFAR-100 dataset exhibit the training latency required for all methods to reach the expected accuracy milestones of [55%, 60%, 65%], which confirms the dominant advantage of our method. It is seen that benchmarks SL, ClassicFL, and FedCS can't reach the expected accuracy milestones of [55%, 60%, 65%] limited to the insufficient training data. Among the HFedCS, HFL, and our method, since HFedCS only selects devices with stronger computational abilities to perform FL training, its accuracy is notably reduced compared to HFL and our method. Furthermore, compared with HFL, our method achieves reductions in training latency of

Table 1: Training latency to achieve expected accuracy milestones on three datasets

Dataset	CIFAR-10			CIFAR-100			SVHN		
Unit	1 ×10^3 s			1 ×10^3 s			1 ×10^1 s		
Accuracy	75%	80%	85%	55%	60%	65%	65%	70%	75%
SL	55	55	55	55	55	55	55	55	55
ClassicFL	116.7	214.9	55	55	55	55	24.0	55	55
FedCS	55	55	55	55	55	55	55	55	55
HFedCS	**16.9**	**24.8**	66.1	**51.3**	55	55	**7.4**	**16.0**	55
HFL	79.6	108.8	204.2	201.6	283.9	477.6	13.2	29.4	91.2
Proposed	20.6	28.0	**56.0**	54.0	**56.6**	**88.3**	9.4	17.2	**46.2**



(a) CIFAR-10 dataset. (b) CIFAR-100 dataset. (c) SVHN dataset.

Figure 4: Accuracy curves on three datasets.

73.24%, 80.06%, and 81.5% to reach three accuracy milestones. These reductions are attributed to our user heterogeneity-aware NN width coefficient determination algorithm, which is well adapted to the computing abilities of heterogeneous devices, thereby eliminating considerable idle time fragments.

7.4 Results on Energy Cost

Table 2: Energy consumption to achieve expected accuracy milestones on three datasets

Dataset	CIFAR-10			CIFAR-100			SVHN		
Unit	1×10^3 J			1×10^3 J			1×10^1 J		
Accuracy	75%	80%	85%	55%	60%	65%	65%	70%	75%
SL	55	55	55	55	55	55	55	55	55
ClassicFL	**30.4**	**56.0**	55	55	55	55	**12.4**	55	55
FedCS	55	55	55	55	55	55	55	55	55
HFedCS	83.5	122.6	326.0	235.1	55	55	20.8	**44.8**	55
HFL	103.7	141.7	266.2	262.9	370.1	622.6	34.0	75.7	234.9
Proposed	85.9	116.7	**233.4**	**224.7**	**235.7**	**367.9**	32.5	59.5	**160.2**

Table 2 exhibits the energy consumption comparison of our method versus the benchmarks in achieving various expected accuracy milestones. The results demonstrate that substantial energy costs can be reduced by our method. Specifically, on the CIFAR-10 dataset, only our method and benchmarks HFedCS and HFL can achieve the highest expected accuracy milestone of 85.0%, and our method can achieve 28.4% and 12.32% energy savings compared to HFedCS and HFL. Similar results are obtained on the CIFAR-100 dataset. For instance, compared to HFL, our method can realize energy reduction of 14.51%, 36.3%, and 40.9% at three expected accuracy milestones, respectively. The primary reasons for energy cost optimization lie in two facts. For one thing, our method assigns subnets with appropriate widths to devices with different computing abilities, so that devices with weaker computing abilities can accomplish their FL tasks of narrow subnets with less resource cost. For another thing, our DVFS-based training energy-saving strategy permits devices to adjust their device operating frequencies to energy-saving gear configurations.

8 Related Work

Recently, researchers have begun to design asynchronous FL training mechanisms to address the system heterogeneity issue. Zhang *et al.* [23] presented an adaptive asynchronous FL training mechanism that can dynamically match the various computing capabilities of devices to adjust their training workloads and shorten the training latency. Wang *et al.* [25] presented a centralized NN model integration scheme to effectively mitigate the performance degradation due to outdated NN models in the asynchronous aggregation. Oftentimes, the above asynchronous mechanisms [23, 25] may affect the training convergence of FL and fail to reach satisfactory precision. Following the synchronous training regime, a greedy user selection mechanism was developed by Nishio *et al.* in [39]. This greedy mechanism speeds up the training by establishing a deadline and selecting as many users as possible who are able to complete FL tasks beforehand. Cui *et al.* [28] proposed an ISODATA-based clustering technique to address the poor training efficiency issue. However, these works [28, 39] overlook the significant constraints imposed by energy consumption of FL training.

In the recent past, many researchers have been devoted to optimizing the FL energy efficiency [26, 27, 41, 42]. Nguyen *et al.* [41] transformed the channel and power decision problem for devices in FL into a solvable stochastic optimization problem, and introduced a reinforcement learning-based resource management strategy for to effectively save energy. Focusing on NN compression, Zhang *et al.* [42] developed an energy-efficient FL framework that jointly adopts pruning, multi-task learning, and unsupervised migration learning to optimize the NN model with respect to model parameters and energy cost. Nevertheless, these studies [27, 41, 42] ignore the computing ability heterogeneity of devices, which may impact the computing efficiency of other devices. To improve FL energy utilization with heterogeneous devices, Zhan *et al.* [26] presented a computing resource management method based on reinforcement learning. However, this mechanism markedly diminishes the contributions of devices with superior computing capacity.

9 Conclusions

To tackle challenges of the system heterogeneity and excessive energy consumption in HFL, this paper proposes a premium multi-width NN-assisted HFL framework. Specifically, we first design a heterogeneity-aware NN width coefficient determination algorithm, which assigns a subnet with the customized width to each user. Then, a width-aware adaptive NN model integration approach is proposed to perform model integration for subnets with different widths. Finally, a latency-aware energy saving approach is introduced to fine-tune devices' operating frequencies. Experiments reveal that, compared with benchmarks, our framework achieves superior accuracy performance and effectively reduce training latency and energy consumption.

Acknowledgements

This work was partially supported by National Natural Science Foundation of China under Grant No. 62272169, No. 62272290, and No.62372184, Natural Science Foundation of Shanghai under Grant No. 24ZR1421500, and Science and Technology Commission of Shanghai Municipality under Grant No. 22DZ2229004.

References

[1] Zhongchi Wang, Hailong Sun, and Zhengyang Zhao. Fedevalfair: A privacy-preserving and statistically grounded federated fairness evaluation framework. In *Proceedings of the ACM International Conference on Multimedia*, page 7191–7199. Association for Computing Machinery, 2024.

[2] Ruixuan Liu, Ming Hu, Zeke Xia, Xiaofei Xie, Jun Xia, Pengyu Zhang, Yihao Huang, and Mingsong Chen. Fedgraft: Memory-aware heterogeneous federated learning via model grafting. *IEEE Transactions on Mobile Computing*, 2025.

[3] Baochen Xiong, Xiaoshan Yang, Yaguang Song, Yaowei Wang, and Changsheng Xu. Client-adaptive cross-model reconstruction network for modality-incomplete multimodal federated learning. In *Proceedings of the ACM International Conference on Multimedia*, page 1241–1249. Association for Computing Machinery, 2023.

[4] Melike Gecer and Benoit Garbinato. Federated learning for mobility applications. *ACM Computing Surveys*, 56(5):1–28, 2024.

[5] Honghong Zeng, Jie Li, Jiong Lou, Shijing Yuan, Chentao Wu, Wei Zhao, Sijin Wu, and Zhiwen Wang. Bsr-fl: An efficient byzantine-robust privacy-preserving federated learning framework. *IEEE Transactions on Computers*, 73(8):2096–2110, 2024.

[6] Wei Wan, Shengshan Hu, Minghui Li, Jianrong Lu, Longling Zhang, Leo Yu Zhang, and Hai Jin. A four-pronged defense against byzantine attacks in federated learning. In *Proceedings of the ACM International Conference on Multimedia*, page 7394–7402. Association for Computing Machinery, 2023.

[7] Md Sirajul Islam, Simin Javaherian, Fei Xu, Xu Yuan, Li Chen, and Nian-Feng Tzeng. Fedclust: Tackling data heterogeneity in federated learning through weight-driven client clustering. In *Proceedings of the International Conference on Parallel Processing*, page 474–483. Association for Computing Machinery, 2024.

[8] Meilin Yang, Jian Xu, Wenbo Ding, and Yang Liu. Fedhap: Federated hashing with global prototypes for cross-silo retrieval. *IEEE Transactions on Parallel and Distributed Systems*, 35(4):592–603, 2024.

[9] Zhiyuan Wu, Sheng Sun, Yuwei Wang, Min Liu, Quyang Pan, Xuefeng Jiang, and Bo Gao. Fedict: Federated multi-task distillation for multi-access edge computing. *IEEE Transactions on Parallel and Distributed Systems*, 35(6):1107–1121, 2024.

[10] Chenrui Wu, Haishuai Wang, Xiang Zhang, Zhen Fang, and Jiajun Bu. Spatio-temporal heterogeneous federated learning for time series classification with multi-view orthogonal training. In *Proceedings of the ACM International Conference on Multimedia*, page 2613–2622. Association for Computing Machinery, 2024.

[11] Ming Hu, Zeke Xia, Dengke Yan, Zhihao Yue, Jun Xia, Yihao Huang, Yang Liu, and Mingsong Chen. Gitfl: Uncertainty-aware real-time asynchronous federated learning using version control. In *Proceedings of IEEE Real-Time Systems Symposium*, pages 145–157, 2023.

[12] Zhuang Qi, Lei Meng, Zitan Chen, Han Hu, Hui Lin, and Xiangxu Meng. Cross-silo prototypical calibration for federated learning with non-iid data. In *Proceedings of the ACM International Conference on Multimedia*, page 3099–3107. Association for Computing Machinery, 2023.

[13] Yichen Li, Wenchao Xu, Yining Qi, Haozhao Wang, Ruixuan Li, and Song Guo. Sr-fdil: Synergistic replay for federated domain-incremental learning. *IEEE Transactions on Parallel and Distributed Systems*, 35(11):1879–1890, 2024.

[14] Shiwei Li, Yingyi Cheng, Haozhao Wang, Xing Tang, Shijie Xu, Weihong Luo, Yuhua Li, Dugang Liu, Xiuqiang He, and Ruixuan Li. Masked random noise for communication-efficient federated learning. In *Proceedings of the ACM International Conference on Multimedia*, page 3686–3694. Association for Computing Machinery, 2024.

[15] Ruofan Jia, Weiying Xie, Jie Lei, and Yunsong Li. Adaptive hierarchical aggregation for federated object detection. In *Proceedings of the ACM International Conference on Multimedia*, page 3732–3740. Association for Computing Machinery, 2024.

[16] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

[17] Lumin Liu, Jun Zhang, S.H. Song, and Khaled B. Letaief. Client-edge-cloud hierarchical federated learning. In *Proceedings of the IEEE International Conference on Communications*, pages 1–6. IEEE, 2020.

[18] Ting Fu, Yu-Wei Zhan, Chong-Yu Zhang, Xin Luo, Zhen-Duo Chen, Yongxin Wang, Xun Yang, and Xin-Shun Xu. Fedcafe: Federated cross-modal hashing with adaptive feature enhancement. In *Proceedings of the ACM International Conference on Multimedia*, page 9670–9679. Association for Computing Machinery, 2024.

[19] Yangguang Cui, Kun Cao, Junlong Zhou, and Tongquan Wei. Optimizing training efficiency and cost of hierarchical federated learning in heterogeneous mobile-edge cloud computing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(5):1518–1531, 2023.

[20] Hengyi Wang, Weiying Xie, Jitao Ma, Daixun Li, and Yunsong Li. Fedsls: Exploring federated aggregation in saliency latent space. In *Proceedings of the ACM International Conference on Multimedia*, page 7182–7190. Association for Computing Machinery, 2024.

[21] Jie Feng, Lei Liu, Qingqi Pei, and Keqin Li. Min-max cost optimization for efficient hierarchical federated learning in wireless edge networks. *IEEE Transactions on Parallel and Distributed Systems*, 33(11):2687–2700, 2022.

[22] Bo Xu, Wenchao Xia, Wanli Wen, Pei Liu, Haitao Zhao, and Hongbo Zhu. Adaptive hierarchical federated learning over wireless networks. *IEEE Transactions on Vehicular Technology*, 71(2):2070–2083, 2022.

[23] Tuo Zhang, Lei Gao, Sunwoo Lee, Mi Zhang, and Salman Avestimehr. Timelyfl: Heterogeneity-aware asynchronous federated learning with adaptive partial training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 5064–5073, 2023.

[24] Liping Yi, Gang Wang, Xiaoguang Liu, Zhuan Shi, and Han Yu. Fedgh: Heterogeneous federated learning with generalized global header. In *Proceedings of the ACM International Conference on Multimedia*, page 8686–8696. Association for Computing Machinery, 2023.

[25] Zhongyu Wang, Zhaoyang Zhang, Yuqing Tian, Qianqian Yang, Hangguan Shan, Wei Wang, and Tony Q. S. Quek. Asynchronous federated learning over wireless communication networks. *IEEE Transactions on Wireless Communications*, 21(9):6961–6978, 2022.

[26] Yufeng Zhan, Peng Li, Leijie Wu, and Song Guo. L4l: Experience-driven computational resource control in federated learning. *IEEE Transactions on Computers*, 71(4):971–983, 2022.

[27] Xinqian Zhang, Ming Hu, Jun Xia, Tongquan Wei, Mingsong Chen, and Shiyan Hu. Efficient federated learning for cloud-based aiot applications. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 40(11):2211–2223, 2021.

[28] Yangguang Cui, Kun Cao, Guitao Cao, Meikang Qiu, and Tongquan Wei. Client scheduling and resource management for efficient training in heterogeneous iot-edge federated learning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(8):2407–2420, 2022.

[29] Luping WANG, Wei WANG, and Bo LI. Cmfl: Mitigating communication overhead for federated learning. In *Proceedings of the IEEE International Conference on Distributed Computing Systems*, pages 954–964, 2019.

[30] Nguyen H. Tran, Wei Bao, Albert Zomaya, Minh N. H. Nguyen, and Choong Seon Hong. Federated learning over wireless networks: Optimization model design and analysis. In *Proceedings of the IEEE Conference on Computer Communications*, pages 1387–1395, 2019.

[31] Jun Xia, Yi Zhang, and Yiyu Shi. Towards energy-aware federated learning via marl: A dual-selection approach for model and client. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 1–9. Association for Computing Machinery, 2025.

[32] Yangguang Cui, Kun Cao, and Tongquan Wei. Reinforcement learning-based device scheduling for renewable energy-powered federated learning. *IEEE Transactions on Industrial Informatics*, 19(5):6264–6274, 2023.

[33] Mingzhe Chen, Nir Shlezinger, H. Vincent Poor, Yonina C. Eldar, and Shuguang Cui. Communication-efficient federated learning. *Proceedings of the National Academy of Sciences*, 118(17):1–8, 2021.

[34] Ming Hu, Peiheng Zhou, Zhihao Yue, Zhiwei Ling, Yihao Huang, Anran Li, Yang Liu, Xiang Lian, and Mingsong Chen. Fedcross: Towards accurate federated learning via multi-model cross-aggregation. In *Proceedings of the IEEE International Conference on Data Engineering*, pages 2137–2150, 2024.

[35] Wenqi Shi, Sheng Zhou, Zhisheng Niu, Miao Jiang, and Lu Geng. Joint device scheduling and resource allocation for latency constrained wireless federated learning. *IEEE Transactions on Wireless Communications*, 20(1):453–467, 2021.

[36] Changsheng You, Kaibin Huang, Hyukjin Chae, and Byoung-Hoon Kim. Energy-efficient resource allocation for mobile-edge computation offloading. *IEEE Transactions on Wireless Communications*, 16(3):1397–1411, 2017.

[37] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *Proceedings of the NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, pages 1–9, 2011.

[38] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *Proceedings of the British Machine Vision Conference*, pages 1–15. British Machine Vision Association, 2016.

[39] Takayuki Nishio and Ryo Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge. In *Proceedings of the IEEE International Conference on Communications*, pages 1–7, 2019.

[40] Jin-Hyun Ahn, Osvaldo Simeone, and Joonhyuk Kang. Wireless federated distillation for distributed edge learning with heterogeneous data. In *Proceedings of IEEE Annual International Symposium on Personal, Indoor and Mobile Radio Communications*, pages 1–6, 2019.

[41] Huy T. Nguyen, Nguyen Cong Luong, Jun Zhao, Chau Yuen, and Dusit Niyato. Resource allocation in mobility-aware federated learning networks: A deep reinforcement learning approach. In *Proceedings of the IEEE World Forum on Internet of Things*, pages 1–6, 2020.

[42] Yu Zhang, Guoming Tang, Qianyi Huang, Yi Wang, Kui Wu, Keping Yu, and Xun Shao. Fednilm: Applying federated learning to nilm applications at the edge. *IEEE Transactions on Green Communications and Networking*, 7(2):857–868, 2023.