FedFold: Efficient Federated Learning for Inconsistent Dual Resource Heterogeneity

Tzu-Hsuan Peng, Chieh-Chi Yang, Chih-Hsin Liu, *Member, IEEE,* Ai-Chun Pang, *Fellow, IEEE,* and Cheng-Wei Huang, *Member, IEEE*

Abstract—With the continuous evolution of 6G networks and edge computing, the demand for intelligent, secure, and efficient distributed learning frameworks is growing. Federated Learning (FL) has become a key approach for privacy-preserving model training, allowing edge devices to collaboratively train a global model without exchanging raw data. However, real-world FL deployments face significant challenges due to system heterogeneity. Devices vary in computational power and network conditions, leading to the straggler effect, where weaker devices slow down training efficiency. While prior works have proposed solutions focusing either on computational or communication heterogeneity, they fail to address the inconsistencies between these two factors, limiting FL's practical scalability. To address this issue, we propose Federated Folding (FedFold), a novel FL framework designed to balance the dual resource efficiency in heterogeneous edge environments. FedFold integrates a progressive model splitting strategy to dynamically distribute training loads and local aggregation-based compression to reduce communication overhead. Unlike existing methods that handle one bottleneck at a time, FedFold adapts to inconsistent and dual resource heterogeneity among clients, ensuring robust and scalable learning. The experiments demonstrate that FedFold significantly reduces training time and communication costs while maintaining high model accuracy, outperforming state-of-the-art FL frameworks by over to 70% in efficiency improvements under non-IID scenarios.

Index Terms—*Federated Learning, System Heterogeneity, Dual Resource Heterogeneity, Model Heterogeneity*

1 Introduction

With the advancements of 6G networks and the rapid development of edge intelligence, there is a growing demand for intelligent services that can process data efficiently and securely. These technologies enable faster data transmission and bring computation closer to data sources, such as smartphones, IoT devices, and autonomous systems. However, processing massive amounts of distributed data while ensuring privacy remains a significant challenge. Federated Learning (FL) emerged as a privacy-preserved solution that allows edge devices to train a global model collaboratively without sharing raw data [1]. Each user trains their data locally and sends model updates to a central server for aggregation. This training scheme enhances privacy and reduces the need for large-scale data transfers.

However, in real-world scenarios, devices often have diverse hardware capabilities and varying network conditions. Some devices have powerful processors and can complete training quickly, while others have limited resources. Similarly, devices with stronger network connections can send updates faster, while those with poor connectivity slow down the process. This leads to the straggler effect, where weak devices drag down the overall training efficiency [2]. As a result, improving the training efficiency is the key to realizing FL for 6G edge computing systems.

Previous studies have addressed the system heterogeneity issues from two separate aspects: **Computation and Communication Heterogeneity**. Most of the existing works focus on tackling the straggler effects caused by computational differences. Solutions like client selection [3], asynchronous updates [4], and model heterogeneity [5] have been proposed, which either prioritize faster devices or adjust model sizes based on device capability. These methods overlooked the potential of weak devices thereby limiting the generalization capability of the global model. For communication challenges, strategies such as model compression and specialized client selection schemes help reduce network load [6] [7]. Applying aggressive compression may not be sufficient to prevent delays if it compromises the model accuracy. However, focusing on only one of these aspects fails to fully solve the straggler effect in highly diverse environments. Moreover, the computation and communication capabilities are often inconsistent among clients in the real world; how to balance the efficiency is one of the main challenges in FL.

Our preliminary experiments reveal that the primary bottleneck for training efficiency in FL varies across devices and situations. The dominant factor affecting training time can be influenced either by computational abilities or communication resources, and the heterogeneities of both factors are often inconsistent. Therefore, it is necessary to address both challenges simultaneously for efficient model training in 6G edge networks. To tackle this issue, we propose a novel federated learning framework designed for heterogeneous edge environments called Federated

Tzu-Hsuan Peng and Cheng-Wei Huang are with the Department of Computer Science and Information Engineering, National Taiwan University, Taipei 10617, Taiwan (e-mail: f09922029@csie.ntu.edu.tw; r10922043@csie.ntu.edu.tw).
Chieh-Chi Yang is with the Department of Computer Science and Information Engineering, National Taiwan University, Taipei 10617, Taiwan (e-mail: b09901055@g.ntu.edu.tw).
Chih-Hsin Liu is with the Department of Computer Science and Information Engineering, National Taiwan University, Taipei 10617, Taiwan (e-mail: b09705024@g.ntu.edu.tw).
Ai-Chun Pang is with the Research Center for Information Technology Innovation, Academia Sinica, Taipei 115, Taiwan, also with the Department of Computer Science and Information Engineering, National Taiwan University, Taipei 10617, Taiwan, and also with the High Performance and Scientific Computing Center, National Taiwan University, Taipei 10617, Taiwan (e-mail: acpang@citi.sinica.edu.tw).

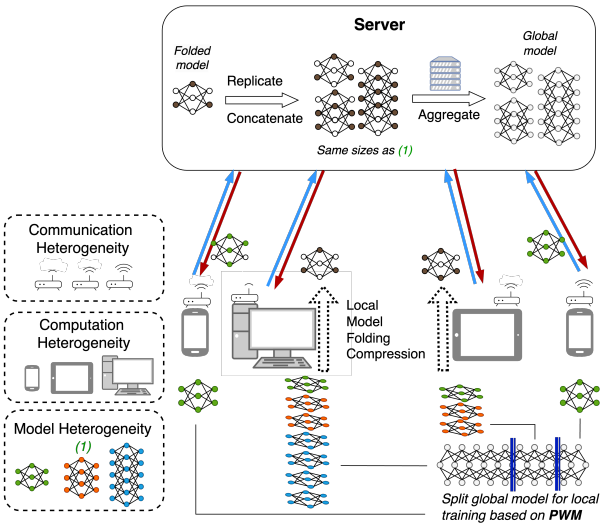


Fig. 1: System Architecture of FedFold in inconsistent computation and communication heterogeneous scenario.

Folding (FedFold). Our previous work, PWM [8], handled the computational heterogeneity issue with model heterogeneity and ensemble technique. Enhanced from our previous work, FedFold integrates the progressive model splitting strategy from PWM to balance the computational load and local aggregation-based compression to reduce communication overhead. As shown in Fig.1, we focus on the inconsistent system heterogeneities among clients. Unlike existing methods that focus on either solely computation or communication heterogeneity, FedFold can balance the training efficiency among clients by dynamically adjusting the training load and minimizing the overall training time. The dual approach ensures efficient and scalable learning across devices with diverse and inconsistent capabilities. Notably, FedFold can surpass all baselines with barely additional overheads in non-IID scenarios.

The contributions of this work are summarized as follows.

- To our best knowledge, we are the first work that considered the system heterogeneity of two dominants in inconsistent scenarios. We found out that both computation and communication can dominate the training time, highlighting the necessity of considering both factors in heterogeneous model training.
- We proposed a novel framework, FedFold, which can address both computation and communication heterogeneity through progressive model splitting and local aggregation-based compression.
- Extensive experiments demonstrate that FedFold reduces training time and communication costs while maintaining high model accuracy. FedFold outperforms all baselines under different device distributions and non-IID degrees. The strategy gains over 70% efficiency improvement compared to the SOTA FL frameworks.

2 Related Work

2.1 System Heterogeneity in Federated Learning

System heterogeneity in FL arises due to differences in computational power and network conditions among participating devices. Traditional FL methods, such as Federated Averaging (FedAvg) [9], assume that all devices have similar capabilities, which is rarely the case in practice. Some devices possess high processing power and complete training quickly, while others struggle due to limited resources. This imbalance leads to the straggler effect, where weaker devices slow down the overall training process. To eliminate the latency caused by device and resource heterogeneity, recent works have mainly focused on two aspects, which are computation heterogeneity and communication heterogeneity.

2.2 Computation Heterogeneity

Computation heterogeneity stems from variations in hardware capabilities, such as CPU performance and memory bandwidth among edge devices. Existing research has focused on mitigating the idle time through client selection, asynchronous updates, and model heterogeneity frameworks.

Client selection is the most intuitive and common technique to improve training efficiency by selecting appropriate clients to participate in the FL system. FedSC [3] actively selects clients with sufficient computational power to tackle challenges in mobile edge computing (MEC) environments. Power-of-Choice [10] addresses selection bias by adjusting the probability of client selection based on dataset distribution, ensuring global model convergence while maintaining balanced client participation. FedAR [11] continuously monitors client activity and local computational resources on IoT devices. By scoring clients based on these factors, it prioritizes those with faster responses, disregarding slower devices to accelerate the training process.

Asynchronous updates enable devices that finish training faster to upload their model parameters without waiting for slower devices. FedAsync [4] introduces a new asynchronous federated optimization algorithm, allowing the server to update the global model as client models are received. SAFA [12] enhances this by performing global updates after receiving a certain percentage of client models, incorporating lag tolerance and a cache structure to minimize waiting time. FedBuff [13] further improves the privacy of asynchronous federated learning by securely aggregating client updates in a buffer before applying them to the global model.

However, both strategies can introduce model bias as stronger devices contribute more frequently to the global model. In a non-IID (non-Independent and Identically Distributed) environment, where data distributions differ across devices, this can skew the global model toward the data distributions of the stronger devices. To address this issue, model heterogeneity approaches have been developed. Model heterogeneity approaches not only help to reduce bias in the global model but also enhance its generalization capabilities, ultimately speeding up convergence times. For example, HeteroFL [5] allows clients to train heterogeneous models locally, which are then aggregated into a single global model. Split-Mix [14] divides models into sub-models based on their width, enabling adaptation to different computational capacities. FedRolex [15] employs a rolling-based sub-model extraction scheme, ensuring that global model parameters are trained more evenly. PWM [8] integrate the model heterogeneity framework with splitting policy and ensemble techniques to mitigate the straggler effect and balance the contribution among devices.

2.3 Communication Heterogeneity

In addition to computation overhead, communication latency significantly impacts the efficiency in FL. Recent surveys highlighted

the variability in communication environments, particularly the differences in bandwidth among clients. FedACG [6] reduced communication costs by transmitting only model parameters, excluding additional information like gradients. Similarly, ZeroFL [16] applies top-K sparsity to models, minimizing the amount of data transmitted while retaining essential neural information. However, these works did not account for variations in network bandwidth across devices, which can lead to inconsistent upload times in federated learning. Some studies attempt to address the communication network heterogeneity challenge through model compression or hierarchical uploading strategies.

A model compression strategy reduces communication overhead by compressing model information, such as updates or gradients, before transmitting them through the network. Compression techniques such as quantization, top-K sparsification, and singular value decomposition (SVD) sparsification help minimize communication overhead by reducing the size of the model. HeteroSAG [17] addresses this by adjusting the quantization level for each client based on their communication capabilities while maintaining secure aggregation. FedCG [7] employs a client selection scheme combined with gradient compression to reduce communication costs, accommodating clients with varying bandwidths.

Other approaches, such as hierarchical uploading, utilize various layers of intermediate base stations or servers to allow clients to aggregate model updates sequentially. For instance, fog learning [18] aims to mitigate communication discrepancies by enabling direct device-to-device (D2D) learning and employing multi-layer aggregation at different scales. In [19], the study addresses heterogeneous cellular networks (HCN) and proposes a hierarchical architecture with macro base stations (MBS) to enable mobile users (MU) to exchange model parameters incrementally.

2.4 Dual Resource Heterogeneity

However, the works mentioned above simplify real-world scenarios by considering only either computation or communication heterogeneity. In reality, both factors significantly impact the training efficiency. FedLamp [20] is one of the few approaches that jointly optimizes computation and communication efficiency. It minimizes communication overhead by dynamically adjusting the local update frequency and model compression rate for each client, optimizing efficiency in heterogeneous networks. Additionally, it takes the varying computational capabilities of devices into account and implements a client selection strategy to prevent training inefficiencies. Unfortunately, FedLamp's reliance on a client selection strategy may introduce a bias toward devices with stronger computational resources and significantly reduce the performance in non-IID scenarios.

3 Preliminaries and System Model

In this section, we demonstrate that both computation and communication time can significantly dominate the training process, depending on the specific configurations of the device and network. We further illustrate that focusing solely on either computation or communication heterogeneity may be inefficient in practical scenarios. Additionally, we found that FedLamp, the only method to date that considers both computation and communication heterogeneity, experiences substantial performance degradation in highly non-IID and system-inconsistent settings. These findings emphasize the need to consider the impact of computation and communication heterogeneity simultaneously to achieve efficient federated learning.

3.1 Preliminary Experiment

We evaluate the impact of computation and communication heterogeneity within our previous PWM [8] framework using the CIFAR10 dataset. According to Speedtest [21] statistics from over 190 countries and more than 11 million daily tests conducted as of August 2024, there are notable variations in both downlink and uplink bandwidths for mobile devices and fixed broadband across different regions. The average uplink bandwidth for mobile devices is 11.02 Mbps, but it can vary from as low as 3.3 Mbps to as high as 28 Mbps. In this experiment, we simulate a dual and inconsistent resource heterogeneous environment. We set the devices with NVIDIA 2080 Ti and Intel Core i5 CPU as the strong and weak computation users. For network conditions, we tested with a 30 Mbps connection to represent strong communication and a 3 Mbps connection to simulate weak communication following the Speedtest statics.

Federated learning faces significant challenges due to the interplay between computation and communication heterogeneity. The results in Table 1 demonstrate that neither computation nor communication consistently dominates the overall training time across all scenarios. For instance, when using an Intel Core i5 with a strong communication network, computation time becomes the primary bottleneck. On the other hand, under weak communication conditions, the communication time significantly impacts training efficiency. This pattern exists across different model architectures, and the phenomenon becomes more severe as the model size increases. These findings emphasize that both computation and communication constraints can independently limit training performance, depending on the specific device and network conditions.

The results in Table 2 show that the SOTA frameworks fail to mitigate the straggler and efficiency issue effectively. Computation-centric approaches, such as HeteroFL [5] and FedRolex [15], aim to address computational heterogeneity by assigning different model sizes to devices based on their capabilities. While these approaches reduce the computation gap, it fails to account for network variations. In cases where communication is slow, high-performance devices remain idle while waiting for updates from weaker devices, limiting the overall efficiency of the system. Moreover, these methods do not mitigate the straggler effect caused by weak communication devices, which prolongs training time despite optimizing computational loads.

Communication-focused methods, such as FedCG [7], attempt to alleviate communication heterogeneity by compressing model updates and reducing transmission overhead. While this approach can balance communication time across devices, it does not resolve disparities in computational power. As a result, devices with limited processing capabilities continue to slow down the training process, as their updates arrive late regardless of communication efficiency. The reliance on communication optimization alone fails to improve overall system performance in scenarios with high computation heterogeneity.

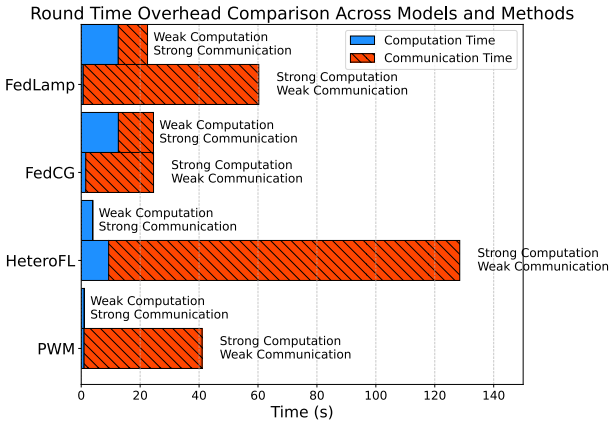
FedLamp [20] is the only framework that takes both computation and communication into consideration. However, its static compression strategy does not dynamically adjust to real-time fluctuations in network bandwidth and device performance. In highly non-IID and inconsistent heterogeneity settings, FedLamp

TABLE 1: Performance comparison of different models on various devices

Model	Device	Network (Uplink Bandwidth)	Computation Time (s)	Communication Time (s)
CNN	GPU NVIDIA 2080 Ti	Strong Comm. (30Mbps)	1.73	0.563
		Weak Comm. (3Mbps)	1.73	5.632
	CPU intel core i5	Strong Comm. (30Mbps)	4.875	0.563
		Weak Comm. (3Mbps)	4.875	5.632
ResNet9	GPU NVIDIA 2080 Ti	Strong Comm. (30Mbps)	3.46	1.759
		Weak Comm. (3Mbps)	3.46	17.598
	CPU intel core i5	Strong Comm. (30Mbps)	14.823	1.759
		Weak Comm. (3Mbps)	14.823	17.598

struggles to maintain efficiency due to its rigid approach. Devices with weaker connections or lower processing power experience excessive delays, leading to imbalances in training. Consequently, FedLamp can not achieve a meaningful improvement over single-focus methods, as it cannot fully mitigate the straggler effect under varying resource conditions.

The persistence of these inefficiencies highlights the limitations of existing federated learning strategies. These findings emphasize the need for a federated learning framework that dynamically optimizes both computation and communication in real-time. Current solutions remain inadequate for mitigating the straggler effect under dual-resource heterogeneity. Without an adaptive mechanism that responds to both computational power and network bandwidth fluctuations, federated learning will continue to suffer from prolonged training durations and inefficient resource utilization. The preliminary experiments prove the limitations of existing work in realistic inconsistent heterogeneity scenarios. This leads us to an important question: ***How can we effectively consider both computation and communication to make the training process more efficient while maintaining accuracy across various environments?***



(a) Round Time Overhead

Fig. 2: Comparison of Round Time Overhead under ResNet18 across Different Methods

3.2 System Model

We consider a typical federated learning scenario which consists of two essential parts: clients and server. First, clients are devices that participate in local training. We denote the set of clients as $C = \{1, 2, 3, \ldots, C\}$, where C is the number of edge devices in each communication round. The total number of edge devices is represented by D, with each device i having a local data distribution X_i. Each client i maintains a model set M_i according to its computational capacity. Device capacities are expressed as ratios relative to the server's performance: $L_i \subset \{1, 0.5, 0.25, 0.125, 0.0625\}$. A value of 1 means the device matches the server's capacity, while 0.0625 represents the weakest. The communication bandwidth for each client i is denoted as N_i, representing The server is a centralized entity with significant computational power, responsible for performing global aggregation. At the end of each communication round, it collects and aggregates all the smaller models uploaded by clients. This process ensures that updates from devices with varying computational capacities are efficiently combined to update the global model. Server maintains a global model set that contains all kinds of split models: $\{1, 0.5, 0.25, 0.125, 0.0625\}$, given that the model with size 1 is W_g.

The computation time of device i can be presented as:

$$T_{i,\text{comp.}} = \frac{t_0 \cdot |X_i| \cdot \text{size}(M_i)}{L_i}, \tag{1}$$

where t_0 is the time per operation, and the communication time for device i is:

$$T_{i,\text{comm.}} = \frac{\text{size}(M_i)}{N_i}. \tag{2}$$

Therefore, the problem we are going to solve is to balance and minimize the total time of each communication round, which can be represented as:

$$\min(T_{i,\text{comp.}} + T_{i,\text{comm.}}), \text{ where } i \in C.$$

Some important notations in this paper are listed in Table 1.

4 Methodology: FedFold Framework

In this section, we propose the Federated Folding (FedFold) framework to balance both computation and communication time among the clients. Traditional methods typically focus on one aspect, either computation or communication. In contrast, FedFold introduces a series of algorithms to efficiently address both challenges simultaneously, ensuring optimal performance across heterogeneous devices and network conditions.

4.1 Progressive Width Splitting

Following our previous work, before training begins, the server initializes a set of models by splitting the global model's width with a shrinking rate $r = 0.5$. The server is assumed to know each client's computational capacity and assigns an appropriate model set to each client accordingly. This ensures that clients with

TABLE 2: Time Comparison on Various Methods on ResNet18

Model	Method	Strong Comp. Weak Comm. (GPU&3 Mbps) Comp time	Comm time	Weak Comp. Strong Comm. (CPU&30 Mbps) Comp time	Comm time
CNN	PWM	0.42	5.632	0.38	0.006
	HeteroFL	1.94	16.62	1.06	0.007
	Fedrolex	0.89	16.62	0.87	0.007
	FedCG	1.01	6.54	5.89	1.66
	FedLamp	0.59	8.31	5.78	1.47
ResNet18	PWM	0.97	40.055	1.03	0.047
	HeteroFL	9.31	119.169	3.88	0.047
	Fedrolex	1.29	119.169	0.99	0.047
	FedCG	1.54	23.06	12.62	11.91
	FedLamp	0.69	59.58	12.58	9.93

TABLE 3: Key Notation Summary

Symbol	Description
C	Number of devices participating in the communication round
D	Total number of all devices
X_i	Data distribution for device i
$M_i = \{m_1, m_2, \ldots, m_n\}$	Model sets of device i
L_i	Computation capacity of device i
N_i	Communication capacity of device i
r	Shrinking rate of splitting width
l_k	Size of layer k
w_k	Weight of layer k
q_i	Quantization level of device i

higher computational power receive larger models, while weaker devices receive smaller ones.

The global model's hidden layers are split into segments proportional to the computational ratios of the devices. Each hidden layer is divided by width, with a shrinking rate of 0.5, aligned to the upper-left side of the global model. The strongest device maintains a model set Mi containing models of sizes $\in \{0.5, 0.25, 0.125, 0.0625\}$, while the weakest device holds only a single model of size 0.0625. This design ensures that each device uses its computational power efficiently.

Given the global model W_g with a width l, it is split iteratively by the shrinking rate $r = 0.5$. Then,

$$ll_i = l \cdot r^i, \quad \text{for } i = 0, 1, 2, \ldots \tag{3}$$

Assume that each client i has computational capacity L_i, expressed as a ratio from the set $\{1, 0.5, 0.25, 0.125, 0.0625\}$. A model set M_i is assigned to client i, containing models with widths proportional to its capacity:

$$M_i = \{m_j \mid m_j = l \cdot r^j,\ r^j \leq L_i\}. \tag{4}$$

4.2 Contribution-Aware Local Aggregation

After receiving the assigned models, clients perform local training on their datasets. Following local training, we apply **contribution-aware local aggregation** before uploading the models to the server. Let the local model set be $\{m_1, m_2, \ldots, m_i\}$ in descending order of width. All models align with the top-left corner of m_1, and the parameters in each layer are aggregated proportional to how many devices contribute to the layer. Specifically, we perform the local aggregation in the following ways:

Each client maintains a set of models $\{m_1, m_2, \ldots, m_n\}$ in descending order of width, the parameter at layer k for model m_j is denoted by $W_k^{(j)}$. Let C_k be the number of devices that contribute to layer k. Devices that do not support certain layers (due to smaller models) contribute a value of zero to those layers. For each layer k, the aggregated parameters W_k^{agg} are calculated as:

$$W_k^{agg} = \frac{\sum_{i \in C} W_k^{(i)}}{C_k} \tag{5}$$

After summing the parameters, it averages each parameter by dividing by the count of contributing models, ensuring that the resulting parameters reflect the average contribution across all input models.

4.3 Local Model Folding Compression

After local aggregation, we "fold" the model into smaller sizes. As depicted in Fig. 3, we first split the hidden layer sizes to their smallest widths and then averaged the hidden sizes of all split models. This process effectively reduces the model size while maintaining performance. From our observations, the output layer of the classification model is crucial, while the other layers show similar performance metrics with minimal differences. This allows us to split and merge the hidden layers without sacrificing overall model effectiveness, as the essential features from the output layer remain intact during the compression process.

Let m represent the model with layers $l_1, l_2 \ldots, l_n$, where l_i is the size of the i-th layer. For hidden layers, define $l_{i,j}$ as the size of the j-th split of layeri such that:

$$l_{i,j}, \text{for } j \in \{1, 2, \ldots, k\}$$

After splitting, compute the average hidden layer size:

$$\bar{l_i} = \frac{1}{k} \sum_{j=1}^{k} l_{i,j} \tag{6}$$

Let O denote the output layer, which remains unchanged, then we have:

$$O = l_{\text{output}} \tag{7}$$

The "folded" model m' consists of the averaged hidden layers and the unchanged output layer:

$$m' = \{\bar{l_1}, \bar{l_2}, \ldots, \bar{l_{n-1}}, O\} \tag{8}$$

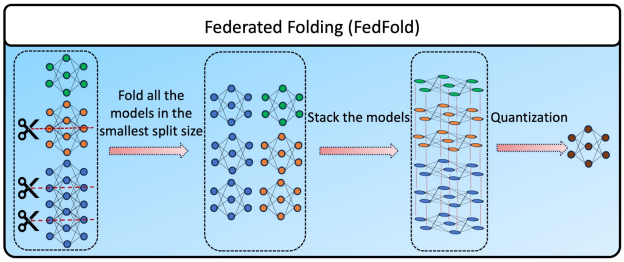


Fig. 3: Local model folding strategy in FedFold.

4.4 Dynamic Quantization

After local aggregation and model folding, all devices have the same model size, which is the smallest model size. However, uploading in poor network environments can cause significant straggler effects, as indicated by preliminary experiments. To address this, we implement dynamic quantization levels for model compression during uploads. Devices in better communication conditions will have higher quantization levels, leading to less compression, while devices with poor network bandwidth will be quantized to a smaller size to meet the constraints. By dynamically adjusting the quantization levels, we ensure optimal preservation of model information while simultaneously reducing communication overhead. This approach allows for efficient utilization of bandwidth, maximizing the effectiveness of model updates based on the communication environment of each device.

The quantized model for client i can be represented as:

$$W_i' = Q(W_i, q_i), \tag{9}$$

where $Q(\bullet)$ is the quantization function, and q_i is the quantization level which is proportional to the network bandwidth:

$$q_i = \alpha N_i$$

4.5 Global Model Expansion Aggregation

After the series of compression, clients upload their models to the server. The server then replicates and concatenates the small model back to its original size. For instance, if a device has an original model set of $\{0.5, 0.25, 0.125, 0.0625\}$, after uploading the model of size 0.0625, the server concatenates this small model into the other three sizes: $\{0.5, 0.25, 0.125\}$, and aggregates all the small models into the global model set.

Let M be the original set of models, where $M = \{m_1, m_2, \ldots, m_n\}$. After compression, let m_{small} be the compressed model size, such that $m_{\text{small}} \in M$. The server concatenates the uploaded model with the remaining models:

$$M' = M \setminus \{m_{\text{small}}\} \cup \{\underbrace{m_{\text{small}}, m_{\text{small}}, \ldots}_{k \text{ times}}\} \tag{10}$$

Finally, define the global model G as:

$$G = \text{Aggregate}(M') \tag{11}$$

The algorithm for the entire training process is as follows:

After training, we adopt a weighted assembly method to ensure accuracy, especially when the ratio of weak devices is high.

Algorithm 1: Local Training

Input: Sorted lists $\{L_i\}_{i=1}^{D}$ and $\{N_i\}_{i=1}^{D}$ representing the computing power and communication capacity of each device. *modelList* is the list of split models.

Output: *modelIndex* sent to the corresponding client for local updating.

Determine device type D_i according to L_i
modelIndex $\leftarrow$ [ ]
if D_i *is a strong device* **then**
  Append each index in *modelList* to *modelIndex*
end
else if D_i *is a weak device* **then**
  Append the index of a BW model in *modelList* to *modelIndex*
end
else
  Sort the list *modelList* in decreasing order by the width
  $mediumBW \leftarrow \lceil \frac{C_i}{C_1} \rceil$
  $index \leftarrow 0$
  while $mediumBW \neq 0$ **do**
    if $mediumBW \geq$ *modelList*$[index]$ **then**
      Append $index$ to *modelIndex*
      $mediumBW \leftarrow mediumBW -$ *modelList*$[index]$
    end
    $index \leftarrow index + 1$
  end
end
Apply local aggregation on *strong device*
if D_i *is a strong device* **then**
  modelIndex, *modelList* $\leftarrow$ *FedFold(modelIndex, modelList) for* D_i
end
Apply Dynamic quantization q_i levels according to the communication capacity N_i. **for** *each device* D_i **do**
  $q_i \leftarrow \alpha \mathrm{N}_i$
  $W_i' = Q(W_i, q_i)$
  $m_q \leftarrow W_i'$
  modelList $\leftarrow [m_q]$
end
Send *modelIndex* to the corresponding client for local updating

5 System Analysis

5.1 Convergence Analysis

To make the system clearer, we can analyze the effect of local aggregation and quantization by the following:

5.1.1 Analysis of Local Aggregation

When devices perform several local updates, model parameters may diverge from the global optimum, especially in non-IID settings, resulting in model drift. The error from local aggregation can be represented as:

$$\Delta W_{\text{global}} = \frac{1}{K} \sum_{k=1}^{K} W_k - W^*, \tag{12}$$

Algorithm 2: FedFold Algorithm

Input: A sorted list *modelIndex* and *modelList* representing the models of device D_i.
Output: *modelIndex* and *modelList* with a folded model.

Aggregate all the models in *modelList*
$W_k^{agg} = \frac{\sum_{i \in C} W_k^{(i)}}{C_k}$
$m_agg \leftarrow W_k^{agg}$
Fold the large model into a small model
for *each layer* l_i *in* m_agg **do**
  $\bar{l_i} = \frac{1}{k} \sum_{j=1}^{k} l_{i,j}$
end
modelList $\leftarrow [m_agg]$
modelIndex $\leftarrow$ [0]

Algorithm 3: Global Aggregation

Input: Lists of models sent from device D_i.
Output: A global model set *G* after global aggregation.

if *model m is sent from strong device* **then**
  M is the original model sets before local aggregation.
  $M' = M \setminus \{m_{\text{small}}\} \cup \{\underbrace{m_{\text{small}}, m_{\text{small}}, \ldots}_{k \text{ times}}\}$
end
Aggregate all the models sent from clients by FedAvg.
$G = \text{Aggregate}(M')$

where K is the number of devices, W_k is the parameter update from device k, and W^* is the true global optimum. Larger divergence ΔW_{global} can slow convergence and affect accuracy. Additionally, local updates impact the convergence rate depending on the number of local steps. The convergence rate bound for local SGD (Stochastic Gradient Descent) with τ steps is given by: $O\left(\frac{1}{KT} + \frac{\tau\Delta}{T}\right)$, where Δ is a measure of gradient dissimilarity among devices, T represents the total number of communication rounds. Larger Δ or greater non-IID data increases this term, slowing convergence.

5.1.2 Analysis of Quantization

Quantization introduces a rounding error to model parameters, which can propagate and affect the model's overall accuracy. You can quantify this error with mathematical metrics:

$$\hat{W} = W + \epsilon_q, \tag{13}$$

where ϵ_q represents the quantization error. $MSQE$ gives a direct measure of the information lost due to quantization and can be computed as:

$$MSQE = \frac{1}{N} \sum_{i=1}^{N} (W_i - \hat{W}_i)^2, \tag{14}$$

where N is the total number of parameters in W. Also, quantization may cause gradient distortion, leading to biased parameter updates. For a given gradient g, the quantized gradient $\hat{g}$ results in:

$$\hat{g} = g + \epsilon_q \tag{15}$$

Convergence can still occur under certain conditions, though slower than full-precision training. Quantization adds an error term $O(\epsilon_q)$ to the standard convergence rate of stochastic gradient descent:

$$E[||\nabla(f(x))||^2] \leq \frac{C}{\sqrt{T}} + O(\epsilon_q), \tag{16}$$

where C is a constant.

5.1.3 Combination Effect Analysis

The combined effect of quantization and local aggregation can be evaluated by combining their respective error terms:

$$O(\frac{1}{\sqrt{KT}} + \frac{\tau\Delta}{T} + \epsilon_q).$$

Also, we observed that the parameters in hidden layers tend to be much smaller than the linear layers, resulting in smaller errors in model drift from local aggregation and distortion from quantization. So the convergence rate can be proximate to $O(\frac{1}{\sqrt{KT}})$, which guarantees convergence when K and T are large enough.

6 Performance Evaluation

We conducted experiments to evaluate our methods, comparing their performance against the state-of-the-art baselines to assess their accuracy, robustness, and adaptability under varying conditions.

6.1 Evaluation Setup

Datasets and Model Architectures: We evaluated our method using two widely used datasets: CIFAR-10 and CIFAR-100. To test the adaptability across different model sizes, we perform our method on CNN, ResNet18, and ResNet152. The experiment included 100 clients, with 10 clients randomly selected to participate in training during each communication round.

Dual Resource Heterogeneity: To simulate diverse device capabilities, we categorized devices into two groups based on their computation capabilities: strong devices and weak devices, assuming a capability ratio of 16:1. The ratio of weak devices is normalized between 0 and 1, referring the portion of weak devices among all clients.For communication networks, we consider four scenarios:

1) Random Fast Comm.: Bandwidth randomly sampled between 10 Mbps and 100 Mbps.
2) Fixed Fast Comm.: Strong devices are paired with weak comm.(10 Mbps) and weak devices are paired with strong comm.(100 Mbps).
3) Random Slow Comm.: Bandwidth randomly sampled between 3 Mbps and 30 Mbps.
4) Fixed Slow Comm.: Strong devices are paired with weak comm.(3 Mbps) and weak devices are paired with strong comm.(30 Mbps).

Random Comm. settings simulate variability, with each client assigned a random bandwidth per communication round. Fixed Comm. assumes extreme cases: strong devices paired with weak comm. (e.g., 3 Mbps for slow, 10 Mbps for fast) and weak devices paired with strong comm. (e.g., 30 Mbps for slow, 100 Mbps for fast). These scenarios validate the adaptability of the methods to both general and extreme cases, ensuring robustness under diverse conditions.

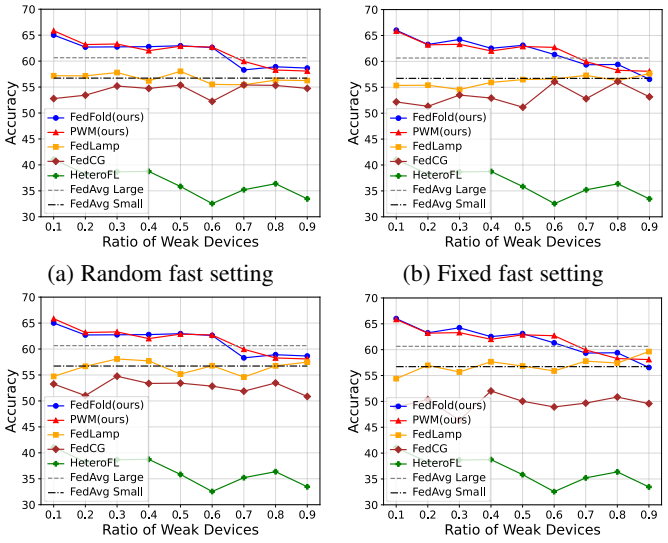
Data Heterogeneity: We follow the non-IID rules outlined in [1] [22] and evaluate our method under varying degrees of non-IID distribution. The degree of non-IID is defined by the number

of classes each client holds. For example, a non-IID degree of 2 means each client is assigned data from only two classes in the dataset.

Baselines: We compare our method with the baselines considering the impact of only computation (PWM [8] and HeteroFL [5]), only communication (FedCG [7]), and both (FedLamp [20]). Additionally, we include FedAvg training with both large and small models that strong and weak devices can afford as general baselines for comparison.

6.2 Model Accuracy Performance

We compared the model accuracy performance of FedFold with baseline methods across different models and datasets under various device ratios. Although results for all architectures are computed, we focus our analysis on the accuracy comparison for ResNet18, as it offers a balanced perspective on model complexity and device heterogeneity. Fig. 4 illustrates the accuracy of ResNet18 across various ratios of weak devices and communication settings. FedFold achieves competitive accuracy compared to PWM and outperforms the SOTA baselines across weak device ratios, demonstrating its ability to handle dual resource heterogeneity effectively. Furthermore, it mitigates the accuracy drop while weak devices dominate. FedCG and FedLamp leverage compression techniques to mitigate the communication overheads, which may loss some information and affect the model performance. By splitting the models before folding, FedFold can prevent the excessive gap among models. With the local aggregation, FedFold can mitigate the communication overhead without discarding any parameters.

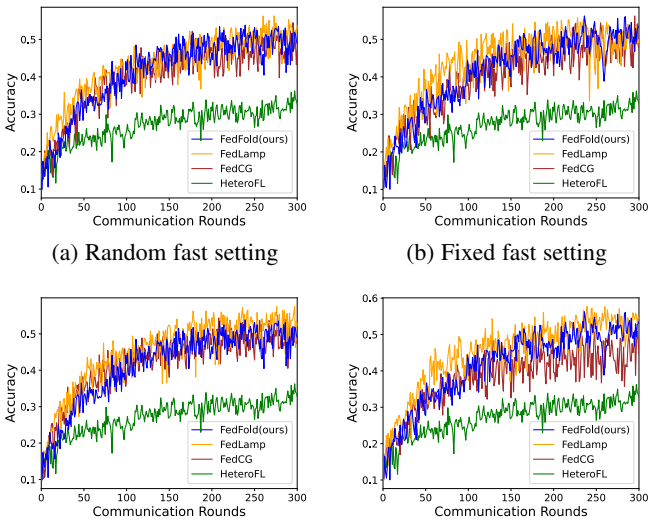


(a) Random fast setting (b) Fixed fast setting

(c) Random slow setting (d) Fixed slow setting

Fig. 4: Global model accuracy of different methods

While FedLamp achieves slightly higher accuracy when the weak device ratio equals 0.9 in slow comm. settings, it benefits from a constant compression rate tied to its local update frequency. This ensures minimal accuracy degradation but does not adequately address the straggler effect or achieve significant time savings. These limitations are explored further in Section 6.3. Moreover, we evaluate our methods with different model architectures and datasets. As shown in Table **??**, FedFold can maintain high accuracy in complicated model architectures and datasets. FedFold is only limited while the model is too simple and small. In such cases, the splitting and folding policies may not provide significant benefits due to the restricted amount of information available. However, in most real-world computational and communication scenarios, the impact of dual heterogeneity is not as severe, making such complex strategies unnecessary.



(a) Random fast setting (b) Fixed fast setting

(c) Random slow setting (d) Fixed slow setting

Fig. 5: Convergence time of different methods

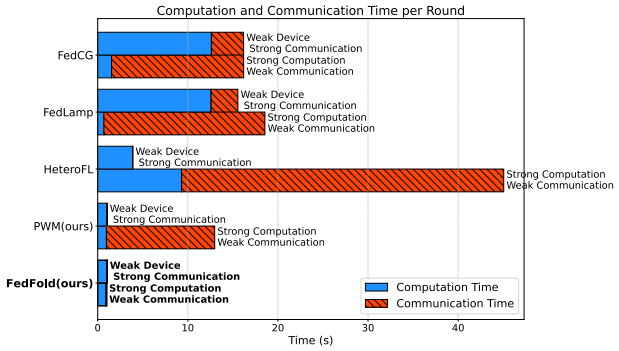
6.3 Wall Time Evaluation

We further evaluate the training efficiency of FedFold and its effectiveness in addressing both computation and communication heterogeneity. As depicted in Fig.5, FedFold can converge slightly faster than other baselines, particularly in fixed comm. scenarios. Furthermore, we analyzed the wall time required to complete a training round under conditions of system heterogeneity. Specifically, we simulated strong devices equipped with GPUs(NVIDIA 2080 Ti) and weak devices using CPU(intel core i5), representing a realistic range of computational capabilities in federated learning setups. As shown in Fig.6, FedFold significantly improves the training efficiency and effectively balances the overall training time among clients.

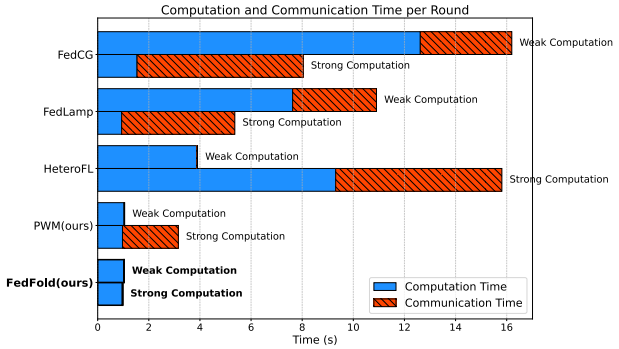
6.4 Ablation Study

We also investigate the impact of individual factors, such as local aggregation extent and quantization level. Aggregation sizes determine how large models running on powerful devices are adjusted to match smaller models during local aggregation. Specifically, Aggregation X1 means that all large models on strong devices are compressed (or ”folded”) to the size of the smallest model. As depicted in Fig. 7, the results indicate that neither aggregation nor quantization significantly impacts model accuracy. This is due to that we do not ”fold” the model weights in the linear layers. The model weights affected by aggregation are less critical than the linear layers, which remain nearly unchanged during local aggregation.

Our approach achieves significant communication savings and model efficiency through local aggregation and quantization strategies. Local aggregation reduces communication costs by 85

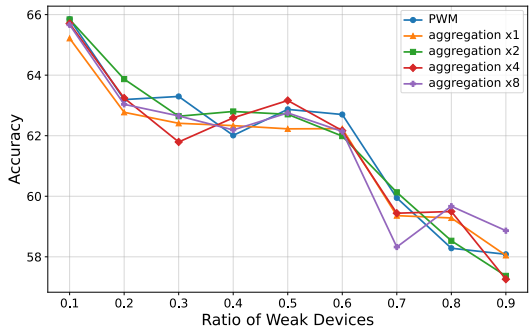


(a) Round Time Overhead with Fixed Fast Comm. Setting

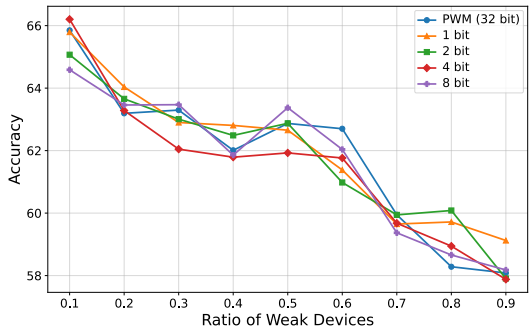


(b) Round Time Overhead with Random Fast Comm. Setting

Fig. 6: Comparison of Round Time Overhead under ResNet18 across Different Methods



(a) Model accuracy over different local aggregation size



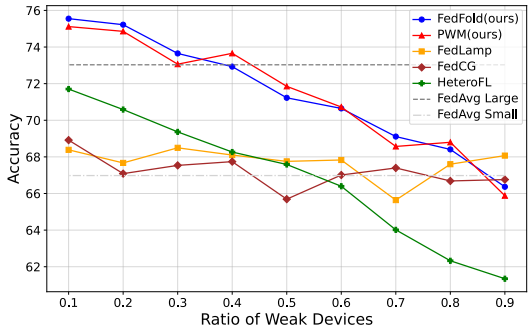
(b) Model accuracy over different quantization level

Fig. 7: Comparison of model accuracy over different factors

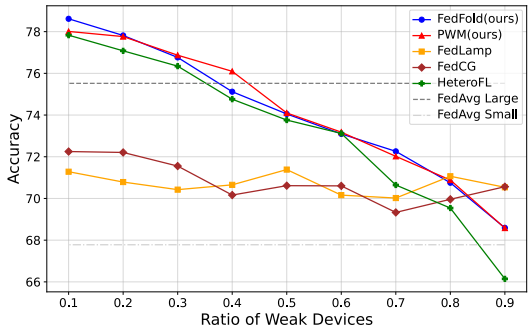
times on large models, making them as compact as their smaller counterparts. Following local aggregation, quantization further enhances efficiency by compressing models up to 16 times based on device-specific communication capabilities. This dual strategy ensures optimized communication without compromising overall model performance.

6.5 Adaptability to Data Heterogeneity

As depicted in Fig. 8, we analyze our method across different non-IID degrees. The setting with a lower degree refers to a more severe non-IID scenario. FedFold performs more stable in different non-IID settings with barely any straggler effects on waiting time. For the same non-IID degree, FedFold can reach a better performance than all baselines. As the non-IID degree becomes more severe, the advantages of FedFold are more obvious. The better generalization performance is inherited from PWM. The different widths of split models can complement each other by extracting different features, and the dynamic weighting policy improved the adaptability by balancing the contribution among devices. Table 4 reveals more details of the adaptability to data heterogeneity with different models.



(a) ResNet18 with non-IID degree 4



(b) ResNet18 with non-IID degree 6

Fig. 8: Model accuracy with different non-IID degree

6.6 Discussions and Limitations

Furthermore, we evaluate our FedFold algorithm under different device compositions, including scenarios with medium devices that possess intermediate computational capacity. Specifically, we analyze two scenarios where the computational capacity of the medium device is 4 times and 8 times that of the weak device, respectively. This evaluation allows us to explore the algorithm's adaptability to a wider range of heterogeneous device configurations, bridging the gap between weak and strong devices in hybrid

environments. The results shown in Fig. 9 demonstrate that our approach is adaptable across different device types and various device ratios.

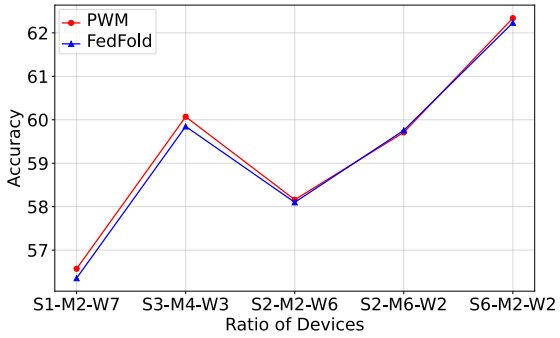
According to our observation, neither aggregation nor quantization has a significant impact on model accuracy if we do not "fold" the linear layers. the weights affected by aggregation are less critical compared to the linear layers, which remain nearly unchanged during local aggregation. Furthermore, this characteristic might be specific to image classification tasks, where non-classification parameters exhibit minimal variation and may not necessarily apply to other types of tasks. Therefore, further investigation is needed to verify the adaptability of this method across different domains.

	Ratio of weak devices	0.7	0.8	0.9
CNN non-IID 4	FedFold	67.48	65.18	61.64
	PWM	68.21	64.96	57.1
	HeteroFL	61.95	60.6	52.89
	FedCG	65.24	65.84	65.45
	FedLamp	72.22	71.82	71.71
	FedAvg(small)	58.5	58.5	58.5
CNN non-IID 6	FedFold	70.98	68.91	63.76
	PWM	71.57	69.47	63.27
	HeteroFL	66.91	65.23	60.6
	FedCG	67.06	66.44	66.42
	FedLamp	74.81	75.05	74.85
	FedAvg(small)	59.57	59.57	59.57
ResNet18 non-IID 4	FedFold	69.11	68.41	66.37
	PWM	68.57	68.79	65.88
	HeteroFL	64.02	62.33	61.35
	FedCG	67.39	66.69	66.76
	FedLamp	65.64	67.6	68.07
	FedAvg(small)	66.98	66.98	66.98
ResNet18 non-IID 6	FedFold	72.26	70.75	68.59
	PWM	72.02	70.88	68.58
	HeteroFL	70.65	69.54	66.15
	FedCG	69.33	69.96	70.56
	FedLamp	70.02	71.07	70.54
	FedAvg(small)	67.78	67.78	67.78

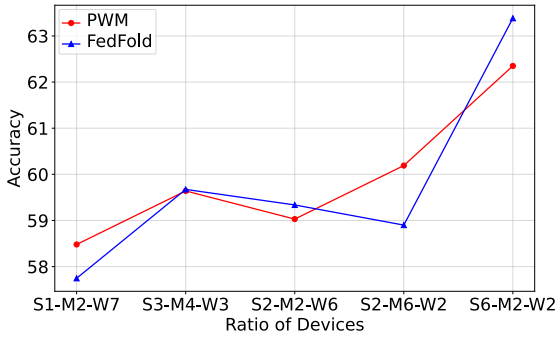
TABLE 4: Model accuracy with non-IID settings of different architectures

7 Conclusion

To address the dual challenges of computational resource heterogeneity and communication bandwidth differences, we propose a novel algorithm, FedFold, which leverages model heterogeneity and local aggregation model compression to efficiently reduce overhead. The local aggregation folding enables all devices to compress their models to the same size, while dynamic quantization adjusts the compression rate of the model uploaded to the server. The global model expansion allows different parts of the global model to be updated without the need for additional model uploads. The two-step compression effectively reduces communication overhead while maintaining model accuracy. Through the experiments, we demonstrate that FedFold outperforms other baselines that focus solely on computation, communication, or both, under various network and non-IID settings. Additionally, we find that performance is largely unaffected by local aggregation, as the hidden layers remain intact during the folding process. In future work, we aim to extend FedFold to a wider range of tasks beyond image classification to further validate its robustness and scalability. This will help overcome its current limitation of being primarily evaluated on image-related tasks.



(a) With medium device size 8



(b) With medium device size 4

Fig. 9: Model accuracy on different device types

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 1273–1282, 2017.

[2] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[3] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," 2019.

[4] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," *arXiv preprint arXiv:1903.03934*, 2019.

[5] E. Diao, J. Ding, and V. Tarokh, "Heterofl: Computation and communication efficient federated learning for heterogeneous clients," *arXiv preprint arXiv:2010.01264*, 2020.

[6] G. Kim, J. Kim, and B. Han, "Communication-efficient federated learning with accelerated client gradient," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, 2024.

[7] Y. Xu, Z. Jiang, H. Xu, Z. Wang, C. Qian, and C. Qiao, "Federated learning with client selection and gradient compression in heterogeneous edge systems," *IEEE Transactions on Mobile Computing*, vol. 23, no. 5, pp. 5446–5461, 2023.

[8] T. H. Peng, C. W. Huang, A. C. Pang, and T. C. Chiu, "Weak devices matter: Model heterogeneity in resource-constrained federated learning," pp. 3274–3279, 2024.

[9] B. McMahan *et al.*, "Communication-efficient learning of deep networks from decentralized data," 2017.

[10] Y. J. Cho, J. Wang, and G. Joshi, "Client selection in federated learning: Convergence analysis and power-of-choice selection strategies," *arXiv preprint arXiv:2010.01243*, 2020.

[11] A. Imteaj and M. H. Amini, "Fedar: Activity and resource-aware federated learning model for distributed mobile robots," 2020.

[12] W. Wu, L. He, W. Lin, R. Mao, C. Maple, and S. Jarvis, "Safa: A semi-asynchronous protocol for fast federated learning with low overhead," *IEEE Transactions on Computers*, vol. 70, no. 5, pp. 655–668, 2020.

[13] J. Nguyen, K. Malik, H. Zhan, A. Yousefpour, M. Rabbat, M. Malek, and D. Huba, "Federated learning with buffered asynchronous aggregation," pp. 3581–3607, 2022.

[14] J. Hong, H. Wang, Z. Wang, and J. Zhou, "Efficient split-mix federated learning for on-demand and in-situ customization," *arXiv preprint arXiv:2203.09747*, 2022.
[15] S. Alam, L. Liu, M. Yan, and M. Zhang, "Fedrolex: Model-heterogeneous federated learning with rolling sub-model extraction," vol. 35, pp. 29 677–29 690, 2022.
[16] X. Qiu, J. Fernandez-Marques, P. P. Gusmao, Y. Gao, T. Parcollet, and N. D. Lane, "Zerofl: Efficient on-device training for federated learning with local sparsity," *arXiv preprint arXiv:2208.02507*, 2022.
[17] A. R. Elkordy and A. S. Avestimehr, "Heterosag: Secure aggregation with heterogeneous quantization in federated learning," *IEEE Transactions on Communications*, vol. 70, no. 4, pp. 2372–2386, 2022.
[18] S. Hosseinalipour, C. G. Brinton, V. Aggarwal, H. Dai, and M. Chiang, "From federated to fog learning: Distributed machine learning over heterogeneous wireless networks," *IEEE Communications Magazine*, vol. 58, no. 12, pp. 41–47, 2020.
[19] M. S. H. Abad, E. Ozfatura, D. Gunduz, and O. Ercetin, "Hierarchical federated learning across heterogeneous cellular networks," pp. 8866–8870, 2020.
[20] Y. Xu, Y. Liao, H. Xu, Z. Ma, L. Wang, and J. Liu, "Adaptive control of local updating and model compression for efficient federated learning," *IEEE Transactions on Mobile Computing*, vol. 22, no. 10, pp. 5675–5689, 2022.
[21] S. G. Index, "Speedtest global index," https://www.speedtest.net/global-index, 2023–2024.
[22] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.