electronics

MDPI

Article

Enhancing Edge-Assisted Federated Learning with Asynchronous Aggregation and Cluster Pairing

Xiaobao Sha [1,2], Wenjian Sun [1,*], Xiang Liu [1], Yang Luo [1,2] and Chunbo Luo [1,2,*]

[1] Yangtze Delta Region Institute (Huzhou), University of Electronic Science and Technology of China, Huzhou 313001, China; xbsha@std.uestc.edu.cn (X.S.); liux@csj.uestc.edu.cn (X.L.); luoyang@uestc.edu.cn (Y.L.)
[2] School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China
* Correspondence: sunwenjian@csj.uestc.edu.cn (W.S.); c.luo@uestc.edu.cn (C.L.)

Abstract: Federated learning (FL) is widely regarded as highly promising because it enables the collaborative training of high-performance machine learning models among a large number of clients while preserving data privacy by keeping the data local. However, many existing FL frameworks have a two-layered architecture, thus requiring the frequent exchange of large-scale model parameters between clients and remote cloud servers over often unstable networks and resulting in significant communication overhead and latency. To address this issue, we propose to introduce edge servers between the clients and the cloud server to assist in aggregating local models, thus combining asynchronous client–edge model aggregation with synchronous edge–cloud model aggregation. By leveraging the clients' idle time to accelerate training, the proposed framework can achieve faster convergence and reduce the amount of communication traffic. To make full use of the grouping properties inherent in three-layer FL, we propose a similarity matching strategy between edges and clients, thus improving the effect of asynchronous training. We further propose to introduce model-contrastive learning into the loss function and personalize the clients' local models to address the potential learning issues resulting from asynchronous local training in order to further improve the convergence speed. Extensive experiments confirm that our method exhibits significant improvements in model accuracy and convergence speed when compared with other state-of-the-art federated learning architectures.

Keywords: federated learning; edge computing; Non-IID data

Citation: Sha, X.; Sun, W.; Liu, X.; Luo, Y.; Luo, C. Enhancing Edge-Assisted Federated Learning with Asynchronous Aggregation and Cluster Pairing. *Electronics* **2024**, *13*, 2135. https://doi.org/10.3390/electronics13112135

Academic Editors: Luis Pires, Ricardo Santos and Joaquim Monteiro

Received: 30 April 2024
Revised: 26 May 2024
Accepted: 28 May 2024
Published: 30 May 2024

1. Introduction

Federated learning (FL) is a distributed machine learning framework for a large number of clients to collaboratively train machine learning models without sharing local data [1]. It is an effective data privacy protection method in the current artificial intelligence environment that prevents the exposure of raw data [2], which has shown extensive applications in many fields, such as healthcare, the intelligent Internet of Things, and finance services [3].

Traditional federated learning is typically structured with a cloud server connecting multiple clients, thus requiring the exchange of a large number of model parameters during multiple update iterations [1]. However, as the amount of participating clients increases, the upstream bandwidth load in the network also increases, thus leading to network congestion and a decrease in the speed of upstream communication [4]. Such communication inefficiency directly affects the overall training performance of the machine learning model [4]. In order to alleviate the huge communication pressure towards the cloud server, existing studies have exploited novel architectures such as adding an edge server layer between clients and the cloud server [5]. The edge server acts as an intermediate in the system by grouping the clients and fusing the machine learning models, which is referred to as edge-assisted FL. By preaggregating local models at the edge layer, the amount of data

updated by the model can be reduced by an order of magnitude [6]. With the same bandwidth, the number of clients occupying the channel is reduced, and the available bandwidth is increased. This approach increases the bandwidth available for each client's upstream link, thereby significantly reducing the time and traffic overhead of transmitting the model parameters from the clients to the server [7]. However, edge-assisted federated learning is constrained by the system heterogeneity, where the clients vary in their computing and communication capabilities; as well, data heterogeneity often happens in different clients with diverse local data. Furthermore, the bottleneck between communication from the cloud to clients is still a pressing issue [6]. For example, during the process of synchronous global model aggregation, the clients remain idle, so a significant amount of waiting time is wasted. While all clients perform asynchronous updates, a rather serious staleness effect occurs if the clients' data are not independently and identically distributed (Non-IID) to a large extent. Therefore, there is a need to develop a set of synchronous–asynchronous hybrid update strategy for edge-assisted federated learning.

In the context of federated learning, client devices are often resource-constrained in wireless networks [8]. Therefore, the system needs to address a multiobjective problem, such as achieving high global model accuracy in the shortest possible time [9]. However, the data owned by each client are Non-IID and influenced by client data drift [10], which significantly slows down the convergence of the global model [11]. Current studies have attempted to solve the FL challenge caused by Non-IID data. Fedprox [12] introduces a proximal term in the loss function to prevent local updates from deviating too far from the initial global model. SCAFFOLD [13] adds a correction term in the local model update formula to overcome gradient disparities. The MOON algorithm [14] draws inspiration from contrastive learning and introduces a model-contrastive loss to ensure the closest possible correspondence between the parameters of the global and local models. These methods effectively alleviate the negative impact of Non-IID data. The three-layer architecture introduces data heterogeneity among the edges, which will be further amplified by the cloud–edge–client model aggregation, thus leading to a decrease in the overall model performance.

Clustered federated learning (CFL) investigates the potential relationships between the data of each client [15] to address the Non-IID problem [16]. In CFL, clients are clustered based on their similarity, and federated optimization is performed within each cluster. This approach greatly improves the performance of the model within each cluster and enhances efficiency [17]. Fundamentally, CFL is closely related to multitask learning: if the tasks within each cluster are similar, they belong to the same learning task. Information exchange between clusters—only occurring during the aggregation process with the cloud server—is limited. Although multitask learning can train models that are highly adapted to each task within a cluster [18], it is difficult to train a global model that can perfectly match all tasks. The aforementioned works have demonstrated a clear correlation between edge–client pairing and CFL. The edge layer in the three-tier FL architecture is paired with a subset of clients to form a cluster. Multiple clusters can then collectively train a global model instead of each cluster training its own separate model. The key challenge is how to maximize the rich and heterogeneous information contained in each cluster to accelerate the global model convergence.

Therefore, we enhance edge-assisted federated learning (EEFL) with asynchronous aggregation and cluster pairing over wireless networks to address the slow convergence and low accuracy of the global model caused by Non-IID training data. In summary, this paper makes the following contributions:

- We propose a novel semi-asynchronous training and aggregation strategy to significantly improve the convergence speed at the edge for the three-layer federated learning architecture. In each training round, the clients in the coverage area of the same edge server are asynchronously trained with the subglobal model obtained after the edge aggregation. This strategy effectively trades the waiting time with local model training to accelerate the convergence.

- We design an edge–client matching strategy to adapt to the semi-asynchronous architecture. Based on the similarity of the local models, the clients are selected to have as far away a distance represented by their data distribution as possible so that the asynchronous training results in each edge cluster are more close to the global model and thus achieve a faster convergence speed.
- We design a new loss function by integrating the concept of personalized federated learning into model-contrastive FL aimed at preserving the historical local information in the local training process, learning both global and local representations, and further accelerating the convergence process to enhance the accuracy of the global model.

The remainder of this paper is organized as follows. Section 2 presents the related work on edge-assisted FL and Non-IID data. Section 3 introduces the edge-assisted federated learning architecture and our motivation. In Section 4, we describe the framework of EEFL and further describe the three innovations. Extensive experiments are conducted in Section 5. Finally, we conclude the paper in Section 6.

2. Related Work

In this section, we review the existing research on federated learning and summarize the differences between our work and the existing work. The survey of existing study focuses on the solution strategies of edge-assisted FL, Non-IID data distribution, and cluster pairing in wireless network scenarios.

2.1. Edge-Assisted FL

In the context of wireless networks, the clients of federated learning are constrained by their geographical locations, thus resulting in dispersed data that require frequent aggregations with the server in order to achieve model convergence [19]. When a large number of devices communicates directly with the cloud through a wide area network, this exacerbates congestion in the backbone network, thus leading to significant communication delays [20]. In order to reduce communication costs and alleviate latency, two approaches have been proposed. The first one is to reduce the amount of data transmitted in each iteration [21], for example, model compression techniques such as network pruning [22] and weight quantization [23]. And the other one is to decrease the communication frequency by selecting client participation in training based on accuracy, data quality, or model update importance [24,25].

Another stream of research is to introduce edge servers as a relay layer in order to mitigate the transmission overhead in the core network [5]. This is achieved by leveraging the higher efficiency of client–edge communication, thus allowing for partial model aggregation. In wireless networks, where the distance between clients and edge servers is generally closer compared to the cloud server, frequent parameter exchanges between edge servers and clients can effectively accelerate the training process. Building upon this, the FedEdge framework [26] further improves communication frequency by asynchronously updating the edge servers with global model parameters received from the cloud server. Nevertheless, directly assigning the received asynchronous global model to the client at the edge does not fully exploit the value of the asynchronous model due to the staleness effect. Differently, our EEFL proposes the exponential moving average (EMA) on the global model during training, which aims to preserve the historical training information.

2.2. Non-IID Data

The challenges faced by federated learning at the communication level primarily stem from the Non-IID nature of the data. The commonly used FedAvg algorithm [1] does not incorporate specific adjustments when confronted with Non-IID data, thus resulting in potential performance losses. Current research in this area primarily focuses on several key aspects aimed at mitigating the impact of Non-IID data on the global model: (1) Improving the loss function for local training: FedProx [12] introduces a proximal term into the loss function, thus ensuring that local updates do not deviate too far from the initial

global model. SCAFFOLD [13] adds a corrective term to the update formula of the local model, thus overcoming gradient differences. Meanwhile, the MOON algorithm [14] draws inspiration from contrastive learning and introduces a model-contrastive loss, thereby maximizing the similarity between the parameter representations of the global and local models. (2) Improving the aggregation method between the server and clients: FedMA [27] utilizes Bayesian nonparametric methods, which identify matching sets of convolutional filters and average them into the global convolutional filters. FedAvgM [28], on the other hand, introduces a momentum mechanism when updating the global model on the server. (3) Personalized federated learning: FedRep [29] divides the model into a base layer and a personalized layer, thus proposing to use the base layer to learn a dimensionality-reduced representation of the global feature representation between data, while the personalized layer serves as a unique local head for each client to achieve personalization. In our EEFL framework, we adopt the idea of personalization and improve the design of the loss function for local training. Compared to the MOON algorithm, our approach, by splitting the model into personalized and shared layers, not only reduces the divergence of model weights during aggregation but also further enhances the local model's ability to extract personalized features.

2.3. Cluster Pairing

To address the heterogeneity of data and device heterogeneity, clustered FL has emerged. In clustered FL, server pair clients based on geographical location, task similarity, data distribution similarity, and resource similarity form clusters [30]. Reasonable grouping and pairing can reduce the differences in system functionality and dataset characteristics among clients within the same group. K-FED [31] completes the cluster of clients with just one round of k means; although it requires fewer communication rounds, the overall model accuracy is lower. FeSEM [32] proposes a multicenter aggregation mechanism, thus utilizing an expectation–maximization method to derive the optimal matching between the clients and the centers. FedGroup [33] groups clients based on the similarity of client update directions and constructs a data-driven distance metric to improve clustering efficiency. Unlike the method of grouping clients with high similarities as mentioned above, HiFlash [6] disperses clients into different edge groups based on different data distributions and response times of the clients, which ensures that the data distribution on the edge nodes is IID. Unfortunately, the HiFlash method needs to collect all the raw label distributions of the clients in the range of edges, which is hard to obtain due to privacy protections. In our work, the cluster pairing only depends on the local models and does not necessitate knowledge of the local dataset features. By computing the similarities between local models and applying three-layer federated learning, the convergence speed of the global model can be improved.

3. Problem Formulation and Motivations

In this section, we first introduce the general problem in three-layer federated learning and then present our motivation of this article through some pre-experiments.

3.1. Problem Formulation

Our focus is to train a machine learning model with the objective of minimizing the sum of losses under the three-layer federated learning architecture.

$$\min_{w} f(w) = \sum_{j=1}^{N} \sum_{i \in \mathrm{C}_j} \frac{|D_i|}{|D|} f_i(w) \tag{1}$$

where N is the total number of edges. We denote C_j as the client set of the jth edge, and the total number of clients can be defined as $K = \sum_{j=1}^{N} C_j$. The size of each local training dateset is D_i. The local loss function of the ith client is

$$f_i(w) = \frac{1}{|D_i|} \sum_{j=1}^{|D_i|} L(w; x_j, y_j) \tag{2}$$

where (x_j, y_j) denotes the input and label of the training sample. $L(w; x_j, y_j)$ is the loss function measuring the difference between the predicted label and true label (usually using the crossentropy loss function).

We consider the cumulative number of communications with the cloud server as the overall iteration count, which is denoted as T.

3.2. Motivation

Our ideas are derived from experiments conducted within the HierFAVG framework [5] using the MNIST dataset [34] as an example. Within this framework, no specific assignment or allocation between clients and edges is performed; instead, a random grouping approach is used. We consider the Non-IID data distribution scenario, where each client possesses only 1–2 label distributions. We draw inspiration from clustered federated learning, which aims to group clients with similar data distributions by exploiting the similarity between local models. This strategy accelerates the convergence of the global model within each group, thus resulting in a higher classification rate for the data labels present in that group. Based upon this, we propose the following hypothesis. Considering that the data distribution among the clients in one group is relatively diversified, and each group contains a substantial number of training samples from different classes, will the training process will be expedited?

To validate our hypothesis, we built a three-layer federated learning system consisting of one cloud server, four edge servers, and 20 client devices. Each client device only possesses samples from 1–2 classes in the MNIST dataset. Three different aggregation strategies are compared:

- Edge-NIID: The data labels owned by clients under each edge are similar (i.e., each edge only contains samples from 3–4 classes).
- Edge-IID: The data labels owned by clients under each edge are different (i.e., each edge contains samples from 8–9 classes).
- Edge-random: The matching between edge servers and clients is performed entirely at random, and the data distribution is random.

As shown in Figure 1, the global model converged slowest in the Edge-NIID scenario, which is clearly unreasonable compared to the other two settings. In three-layer federated learning, the information that the client can interact with is largely derived from the client information under its paired edge coverage. If this portion of information tends to be consistent, it will not be conducive to training a comprehensive model. On the other hand, the convergence speeds of the Edge-IID and Edge-random strategies are similar, thus achieving high accuracy. However, this does not imply that the random matching approach is optimal. The pre-experiments only consider the data distribution among clients while ignoring many other aspects like the data amount and data quality. In the following sections, we will extend this idea and explore a novel edge-assisted federated learning scheme designed to ensure more efficient convergence.

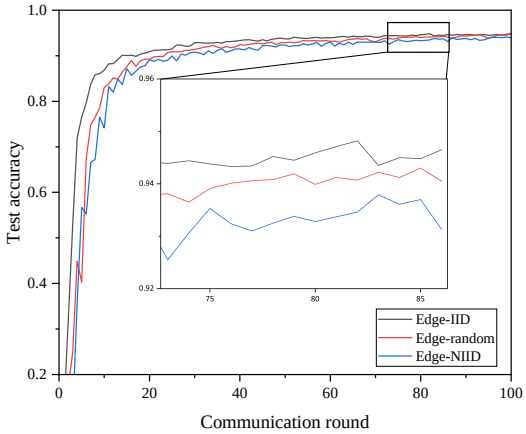


Figure 1. Pre-experiments on three different aggregation strategies on MNIST dataset.

4. Method

In this section, we first propose a scheme to enhance edge-assisted federated learning, which is named as EEFL. Then, we sequentially introduce three innovative aspects of the scheme, namely crosslayer asynchronous aggregation, data feature extraction, and improved loss function.

4.1. EEFL Design

Our study proposes a new edge-assisted federated learning scheme to address the slow convergence and low global model accuracy issues caused by Non-IID data in existing FL architectures. The basic architecture is illustrated in Figure 2.

In the edge-assisted federated learning architecture, there are three layers. The first and the third layer correspond to the client layer and the server layer, respectively, in the traditional FL architecture. The HierFAVG algorithm [5] explores the concept of edge-assisted federated learning, which involves the integration of an edge server (referred to as "edge") into the intermediate layers of the FL framework. In this system, we assume the presence of K client devices, denoted as $Users = \{u_1, u_2, \cdots, u_K\}$, N edge servers for intermediary communication, symbolized as $Edges = \{e_1, \cdots, e_N\}$, and a cloud server C. The specific training process is as follows:

Step 1: Local training on clients. Client devices train their local models on their respective datasets for a certain number of iterations (denoted as τ_1) and subsequently transmit these local models to their corresponding edge servers.

Step 2: Edge aggregation. Each edge server generates a subglobal model by aggregating the local models from its associated clients. If the predetermined rounds (τ_2) have not been reached, the edge server distributes this model to its matching clients. Otherwise, the aggregated model will be transmitted to the cloud server.

Step 3: Cloud aggregation. The cloud server generates a global model by aggregating the submodels received from the edge servers.

Step 4: Model updates. The cloud server sends the aggregated global model to the edge servers, which in turn transmit the global model to the client devices. Finally, the clients use this global model as the initial local model for the next round of training, thus updating their local models accordingly.

Building upon the above algorithm, we proposed a data feature mining mechanism and a new three-tier federated contrastive learning framework. The cloud server explores the data distribution characteristics of client devices based on model parameters and groups client devices with edge severs according to their data distribution patterns. During the local update process on client devices, contrastive learning is employed to further retain personalized information. This process continues until the accuracy of the global model reaches a predetermined threshold.

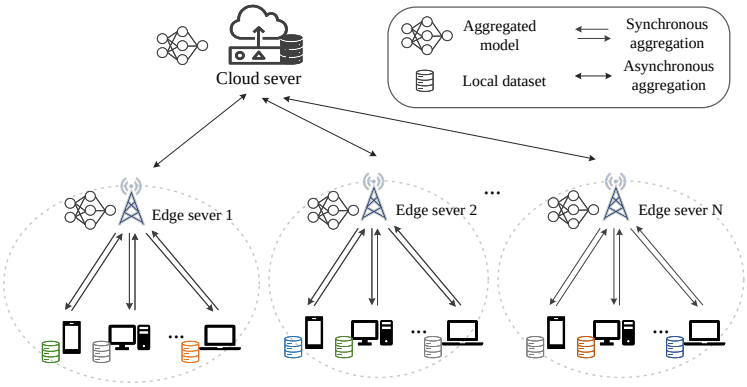


Figure 2. Framework of EEFL. To leverage the idle time of clients during the communication between edges and cloud server, in each round of training, the edge models obtained by edge aggregation are used to train the clients asynchronously.

4.2. Crosslayer Asynchronous Aggregation in Federated Learning

In the current workflow of edge-assisted federated learning, the round trip time (RTT) from edge to cloud server is usually longer than that from the client to the edge (The RTT between client and the cloud server is 150 to 300 ms, while the RTT between the client and the edge server is much lower, ranging from 10 to 40 ms, which is similar to the latency performance of commercial 5G networks.). The convergence speed would be significantly impacted if all aggregations are performed synchronously. During the process of aggregating local models from clients at the edge, client devices remain idle waiting for the cloud server to complete aggregation and distribution of the global model. The vanilla federated learning framework would work ideally if the network conditions remain stable, and thus, all participants, including client devices, edge servers, and cloud servers, consistently execute their corresponding tasks without any delay. However, in practical wireless networking scenarios, network conditions vary, and participant responsiveness may differ. This inefficient aggregation process could result in substantial time waste throughout the entire system.

Therefore, our study introduced a crosslayer asynchronous aggregation mechanism into the existing three-layer synchronous FL architecture, thus aiming to save time and accelerate convergence.

Specifically, after the edge sends the intermediate model $w_{e_i}^t$ to the cloud server C, the edge server will immediately send $w_{e_i}^t$ to its paired clients for local federated training. The main process is equivalent to adding Step 3-1 in parallel to Step 3 in 3-layer FL as follows:

The client u_j updates the local model $w_{u_j}^t$ with $w_{e_i}^t$, thereby training separately on their dataset. Then, the local model is transferred to the pairing edge sever e_i. This process is repeated multiple times from Step 1 to Step 2 until the cloud server sends the global model w_c^t to all edge servers. At this stage, the edge devices reaggregate the global model with the subglobal models:

$$w_{e_i}^t = \alpha w_c^t + (1 - \alpha) w_{e_i}^t \tag{3}$$

After the aggregation is completed, the updated global model $w_{u_j}^t$ is sent to each client for the next round of reaggregation.

4.3. Data Distribution Feature Extraction

The primary goal of federated learning is to train a global model with high accuracy in the shortest possible time. However, the Non-IID nature of data among clients can significantly reduce convergence efficiency. Clustered federated learning addresses this issue by partitioning clients into groups based on their similarity. By performing federated optimization within each group, CFL can greatly improve the model's performance within

each group and enhance the overall efficiency. However, as evidenced in the experiments shown in Figure 1, when the data distribution is scattered and each client only possesses one or two labels, clustering similar clients into groups slows down the convergence speed of the global model, even though the aggregated global model within each group achieves higher recognition accuracy for that specific label. When similar clients are clustered into one cluster and optimization is only performed on the models within the cluster, the variety of data labels is limited, thus making it difficult to train a comprehensive global model.

Based on this premise, the clients were initially grouped based on their local models' similarity. The goal was to ensure that the data distribution within each cluster encompassed as much characteristics of the entire dataset as possible. (i.e., client distributions within clusters are dissimilar, while those between clusters are similar). We considered a cluster as the coverage of an edge server. In conjunction with three-tier federated learning, we designed the algorithm below to improve the accuracy of the final global model and accelerate the convergence speed.

To measure the similarity of the clients and avoid excessive computational overhead, we employed Singular Value Decomposition (SVD) on specific layers of local model. The singular values Σ^i obtained from the decomposition serve as the model's features and are reduced to a one-dimensional array. It is recommended to use a specific layer close to the input layer for analysis. Subsequently, we calculate the cosine similarity between different groups of clients based on the singular values Σ^i, which is written as

$$sim(\Sigma^i, \Sigma^{i'}) = \frac{\left\langle \Sigma^i, \Sigma^{i'} \right\rangle}{\left|\Sigma^i\right| \cdot \left|\Sigma^{i'}\right|} \tag{4}$$

and thereby, we can construct a cosine similarity matrix S. Using the similarity matrix S, we group the clients based on their similarity, thus dividing them into K/N groups arr. Starting from the first client, we select K clients for each group in descending order of similarity until all the groups are formed (ensuring that clients within each group have similar data distribution characteristics). Each edge server randomly pairs with one client from each group, thus resulting in N pairs (where the data distributions of clients within each pair are dissimilar). The specific approach is given in Algorithm 1.

Algorithm 1: Data feature extraction

Input: Local model of client i on the t-round update w_t^i, $i \in \{1, 2, \cdots, K\}$
Output: Client set that each edge is paired with C_j, $j \in \{1, 2, \cdots, N\}$.

Construct a client set $ClientSet = \{1, 2, \cdots, K\}$;
for $i \in \{1, 2, \cdots, K\}$ **do** `// compare client similarity`
  $w_t^i = part(w_t^i)$;
  $\Sigma^k = SVD(w_t^i)$;
  $\Sigma^k = \Sigma^k.resize()$;
end
for $i \in \{1, 2, \cdots, K\}$ **do** `// construct similarity matrix`
  for $i' \in \{1, 2, \cdots, K\}$ **do**
    $S[i][i'] = sim(\Sigma^i, \Sigma^{i'})$
  end
end
for $i \in \{1, 2, \cdots, K\}$ **do** `// assign clients to edge`
  if *client* $i \notin ClientSet$ **then**
    continue
  Choose the top-K indexes both in sorted $S[i]$ and in $ClientSet$;
  Construct the $arr[j]$, which equals to the chosen indexes above ;
  Remove the chosen indexes from $ClientSet$;
end
for $j \in \{1, 2, \cdots, N\}$ **do**
  Randomly select an unselected client from each row in arr to form C_j;
end

4.4. Improvement on Model-Contrastive Federated Learning

In this part, we propose incorporating personalized federated learning into model-contrastive FL. By simultaneously learning global representations and local representations, we aim to enhance the FL global model performance.

The existing methods commonly encourage maintaining consistency between the local models on the clients and the global model in federated learning, thereby overlooking the heterogeneity and individuality of client data. Our method is inspired by the MOON algorithm, which improves the test accuracy by introducing consistency between the representations of local models and the global model. However, it also brings forth new challenges: (1) The computational burden on the clients significantly increases, as they need to compute additional representations of the same input on the global model and the previous round's local models. (2) The previous round's local model is reassigned, thus representing only the local learning status of the preceding round and failing to reflect the client's learning progress in previous rounds. (3) During local updates on the clients, the algorithm fails to explore the client's own local model features, thus leading to the loss of data features that are more useful for the local task.

In this section of the study, we aim to solve the above challenges and further modifies the existing FCL model as illustrated in Figure 3. Following the principles of federated personalized learning, we divided the training model into a shared layer w_{shared} close to the input and a personalized layer w_{per} close to the output. On the one hand, we compared the representation of the shared layer in the local model z_{l_s} with the representation of the shared layer in the global model z_{g_s}. On the other hand, we compared the personalized representation of the current round's local model $z_{l_s}^t$ with the accumulated personalized representation of the historical local models $z_{l_s}^{t-1}$ obtained through Exponential Moving Average (EMA). Therefore, the loss function in this paper will be divided into two parts. The first part is the supervised learning classification loss l_{sup}, which is the crossentropy loss. The second part is the model comparison loss l_{con} mentioned above, which is written as

$$l_{con} = \mu l_{con1} + \gamma l_{con2} \tag{5}$$

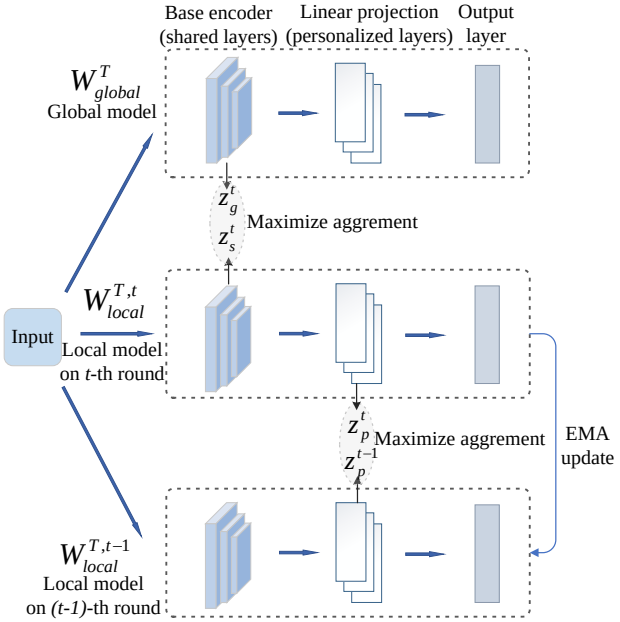


Figure 3. Design of model−contrastive federated learning loss.

In Equation (5), μ and γ represent the weights for the two types of comparison losses individually. The first loss l_{con1} represents the maximization of consistency between the representations of the shared layers in the local model and the global model.

$$l_{con1} = -\log \frac{\exp(sim(z_s^t, z_g^t)/\tau)}{\exp(sim(z_s^{t-1}, z_g^t)/\tau) + \exp(sim(z_s^t, z_g^t)/\tau)} \tag{6}$$

where z_g^t and z_s^{t-1} are the positive sample pairs and negative sample pairs of z_s^t .

l_{con2} represents the maximization of consistency between the representations of the personalized layers in the local model between the current round and the historical rounds.

$$l_{con2} = \left\| \frac{z_p^t}{\left\| z_p^t \right\|} - \frac{z_p^{t-1}}{\left\| z_p^{t-1} \right\|} \right\|^2 \tag{7}$$

Here, the historical models are updated using the EMA method.

$$w_{local}^{t-1} = \begin{cases} \alpha w_{local}^{t-1} + (1-\alpha) w_{local}^t, t > t_{EMA} \\ w_{local}^t, t \leq t_{EMA} \end{cases} \tag{8}$$

When the global update rounds have not reached the threshold t_{EMA}, the local historical model is assigned in an one-time manner, thus indicating that the EMA update method is not beneficial for the learning of the local model at this stage. Once the global round exceeds the threshold t_{EMA}, the local historical model will start to remember the historical models using the EMA method. The EMA approach allows the historical model of clients to comprehensively represent the information from all its previous rounds, rather than just the information from the previous round alone.

5. Experiment

In this section, we present simulation results for our EEFL method to illustrate the advantages of the new three-layer FL system.

5.1. Experiment Setup

In order to evaluate effectiveness of the proposed algorithm EEFL, we conducted comprehensive experiments in a simulated wireless network environment consisting of 20 clients, four edge severs, and a cloud sever. We used three real-world datasets to evaluate the performance of our algorithm, including MNIST, CIFAR10, and FashionMNIST:

- MNIST [34]: The MNIST dataset consists of 70,000 grayscale images of handwritten digits, with each sized 28 × 28 and divided into 10 classes. Each class contains 7000 images. The dataset is further divided into 60,000 training images and 10,000 test images.
- CIFAR10 [35]: CIFAR10 is a dataset of 60,000 color images, with each being 32 × 32 pixels in size and divided into 10 classes. It has 6000 images per class, with a training set of 50,000 images and a test set of 10,000 images. Unlike the grayscale images in MNIST and FashionMNIST, CIFAR10 contains color images, which adds an additional dimension to the data.
- FahionMNIST [36]: This dataset consists of 70,000 grayscale images of fashion products, with each image sized 28 × 28. The dataset is categorized into 10 different categories, with 7000 images in each category. Similar to the MNIST dataset, FashionMNIST also includes a training set of 60,000 images and a test set of 10,000 images.

To fully evaluate the proposed federated learning algorithm, the experiments adopted multiple neural networks. For MNIST, we used LeNet as the model trained on the clients. For CIFAR10, we used a CNN network as the base encoder, which has three 3 × 3 convolution layers followed by 2 × 2 max pooling and fully connected layers with ReLU activation. For FashionMNIST, we used CNN as the base encoder, which has two 3 × 3 convolution

layers followed by 2 × 2 max pooling and fully connected layers with ReLU activation. Note that all baselines used the same network architecture as OURS for fair comparison. The detailed setup are displayed on Table 1. The hyperparameters μ and γ can be determined by grid search in practice.

Table 1. Experiment details on different datasets.

	MNIST	CIFAR10	FashionMNIST
batch size	20	25	32
learning rate	0.1	0.01	0.1
hyperparameter μ	10	0.05	0.01
hyperparameter γ	0.1	10	1
total parameters	21,840	3,493,197	421,642

To simulate the data heterogeneity among clients similar to practical scenarios, in addition to the common IID setting, we employed different data distribution strategies for different datasets. For MNIST and FashionMNIST, the samples in each client were drawn from one to two randomly selected classes. For the CIFAR10 dataset, we utilized the Dirichlet distribution [37] to generate Non-IID data partitions among participants. We sample $p_k \sim Dir_N(\beta)$ and allocate the proportion $p_{k,j}$ of class k instances to participant j, where $Dir(\beta)$ is a Dirichlet distribution with concentration parameter β, which is defaulted to 0.5. With the aforementioned partitioning strategy, each participant may have relatively few (or even no) data samples in certain classes.

5.2. Evaluation and Baseline

Evaluation Metrics. We used three metrics to comprehensively evaluate the performance of our algorithm:

- Test accuracy: The top accuracy of the global model after 200 communication rounds.
- Rounds with the cloud sever: Communication rounds to achieve the target accuracy.
- Time and energy consumption: Total time and energy consumed to achieve target accuracy.

$$T_{comp} = \frac{cD}{f}, T_{comm} = \frac{M}{Blog_2(1 + \frac{hp}{\sigma})} \tag{9}$$

$$E_{comp} = \frac{\alpha c D f^2}{2}, E_{comm} = pT_{comm} \tag{10}$$

Baseline models. We compared our proposed algorithm with both traditional centralized methods and federated learning schemes for performance evaluation:

- Centralized learning: This scheme collects all the raw data to the cloud for training and provides an upper bound for model accuracy.
- FedAVG [1]: A traditional cloud-based FL scheme with synchronous aggregation.
- FedProx [12]: A cloud-based FL scheme with synchronous aggregation, which adds a proximal term in the local training process.
- MOON [14]: A cloud-based FL scheme with synchronous aggregation, which adds extra model-contrasitive loss in the local training process.
- HierFAVG [5]: A cloud–edge–client hierarchical FL scheme that performs synchronous update in both client–edge aggregation and edge–cloud aggregation.

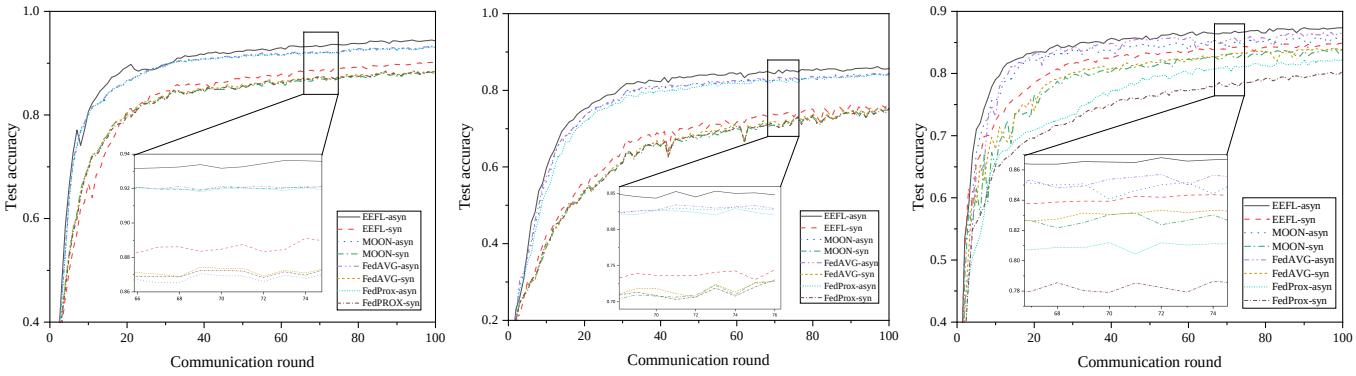
It is worth noting that we adopted random client–edge pairing for the traditional two-layer FL schemes (e.g., FedAvg, FedProx, and MOON) to extend to three-layer FL for fair comparison. In addition, other algorithms were trained from scratch. However, since this algorithm performs pretraining on the client side for similarity grouping, in order to ensure fairness, the same number of pretraining rounds were deployed for other algorithms as well. The update strategies were divided into synchronous and asynchronous types for fair comparison.

5.3. Experimental Results

5.3.1. Test Accuracy and Iterations

In this section, we evaluate the final accuracy of the global model and global iterations with the cloud sever under different baselines both in synchronous and asynchronous aggregation. To mitigate costly communication with the cloud, local computations performed by three-layer federated learning within a global round hugely increase. Hence, we assessed the test accuracy in relation to the overall number of training epochs on the server. By default, we performed 20 rounds of local self-training and 2 rounds of client–edge parameter communication within each round. To accelerate the convergence speed, we doubled the number of local self-training rounds to 40 for the first 30 global rounds of each algorithm. In the later stages, we reduced it to 10 rounds.

Figure 4 illustrates our experiments on the aforementioned three datasets, thus examining the model accuracy and computational efficiency across various training approaches. We categorized the update modes of the edge server into two types: asynchronous aggregation (ours) and traditional synchronous aggregation, where the clients and edge servers remain silent until they receive the global model initiated by the cloud server. To ensure fair comparison, we deployed these two update modes, which are referred to as *asyn* and *syn*, in each benchmark where modifications were made to the local loss function.



(1) MNIST (2) CIFAR10 (3) FashionMNIST

Figure 4. Test accuracy of different FL methods under synchronous and asynchronous update.

According to Figure 4, we can observe that, regardless of the federated learning algorithm used, the aggregation speed in the asynchronous mode was significantly faster than that in the synchronous mode. It is proved that our asynchronous approach saves unnecessary waiting time and enables the entire system to reach the target accuracy faster. Specific data can be obtained from Table 2, where experiments on the three datasets are shown. During the specified 100 or 200 communication rounds with the cloud server, our algorithm achieved higher top accuracy compared to the existing state-of-the-art federated learning algorithms, with an improvement of approximately 6% to 10%. Even when other algorithms were deployed in the asynchronous federated learning framework, our approach still outperformed them by around 2%. In addition, by comparing the required communication rounds to achieve target top accuracy, our proposed method consistently demonstrated excellent performance across different datasets. It could achieve the specified accuracy with approximately 60% to 80% fewer communication rounds. Compared with the current three-layer federated learning schemes, our method achieved a 2.5×–7.0× speedup in acceleration.

Table 2. The top accuracy in 100 rounds with cloud server and the communication rounds to achieve target accuracy.

		EEFL (OURS)		FEDAVG [1]		MOON [14]		FEDPROX [12]	
		Top Acc	Rounds to Target	Top Acc	Rounds to Target	Top Acc	Rounds to Target	Top Acc	Rounds to Target
MNIST	asyn	0.9446	30	0.9328	33	0.9309	31	0.932	32
	syn	0.9043	85	0.8831	100+	0.8832	100+	0.8836	100+
Cifar10	asyn	0.862	29	0.8443	35	0.842	33	0.8417	44
	syn	0.769	161	0.7572	171	0.7526	165	0.7523	177
FashionMNIST	asyn	0.8736	31	0.8579	61	0.8568	76	0.8232	100+
	syn	0.8487	113	0.8441	127	0.8335	139	0.8034	200+

The target top accuracy of MNIST dataset is 90%, and the target top acc of Cifar10 and FashionMNIST is 85%.

5.3.2. Similarity Pairing Strategy

In this section, we conducted experiments on the strategy of similarity pairing. We primarily evaluated our similarity-based pairing method with the HiFlash [6] method and random pairing method when other conditions were kept the same. Since our article focuses on data heterogeneity without considering response latency, HiFlash, for completing the pairing, only utilizes the clients' label distributions to calculate JS divergences. Both our method and HiFlash are dedicated to making the data distribution at the edge different, namely Edge-IID. The main experiments were conducted on the CIFAR10 dataset with the same parameters as mentioned earlier. The experimental results are shown in Figure 5. Our pairing strategy performed much better than the other strategies, thus achieving 1.5% to 5% higher top accuracy.

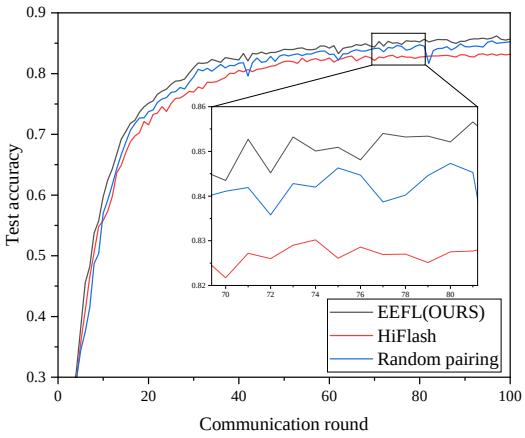


Figure 5. Comparison among different pairing strategies on CIFAR10 dataset.

As a result, we validate the hypothesis from the pre-experiment in Section 3 that the aggregation effect of Edge-IID we proposed is higher than Edge-random through the optimization of the loss function and asynchronous updates strategy. When the client data distributions covered by each edge sever were farther apart, the asynchronously updated subglobal models at the client–edge were found to learn more data features. Consequently, when the edge servers aggregated parameters to the cloud server, the cloud server also learned more data features, thus facilitating faster convergence of the global model.

The reason why HiFlash performed poor compared to the other two methods is that it does not constrain an equal number of clients under each edge. Although it directly ensures that the data distribution under each edge approximates Edge-IID, in the new framework we proposed, the asynchronous aggregation phase provides the edge models with more training opportunities. Unequal client numbers at the edge will pose significant challenges to the performance of the global model. Furthermore, our method does not

require obtaining the label distribution of clients in advance; instead, it directly pairs through the models' partial layers. Hence, our EEFL approach is safer and also facilitates extension to other types of datasets.

5.3.3. Loss Function

In this part, we conducted ablative experiments about the loss function to evaluate the effectiveness of this module. According to the Equation (4), our new loss function has three parts: crossentropy loss and model-contrastive loss in shared layers and in personalized layers. To measure the effectiveness of these parts, we conducted the experiments below in the edge–cloud asynchronous update mode. We fixed the hyperparameters μ and γ with the best-performing values in the loss function and changed the different combination to validate the effectiveness of our proposed algorithm.

The list of the different loss function designs we need to compare includes the following:

- Our design.
- Crossentropy loss and the first comparison loss.
- Crossentropy loss and the second comparison loss.
- Crossentropy loss only.

From Figure 6, our designed loss function led to an improvement of approximately 2% to 5% in the accuracy of the global model. Among them, the loss function that did not utilize model-contrastive learning had the lowest accuracy, which was equivalent to FedAvg. The first contrastive loss only considered the lower level of the model (i.e., parameters capturing common features in images), while the second contrastive loss only considered the upper level of the model (i.e., parameters capturing fine details in images). Considering each aspect separately without their combined use will result in a decrease in accuracy; our algorithm combined the advantages of these two contrastive losses and captured the commonalities among Non-IID clients while preserving the individuality of each client's data, thus resulting in the highest accuracy.

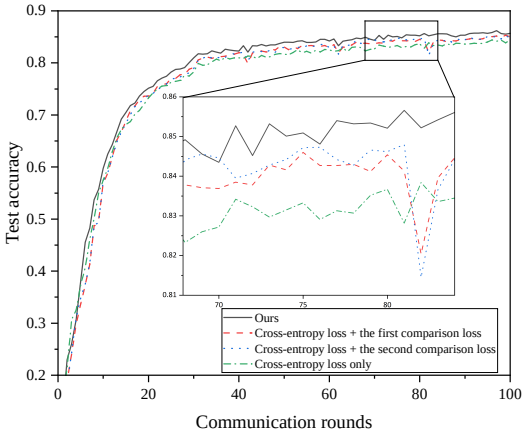


Figure 6. Comparison between different loss design on CIFAR10 dataset.

The reason behind this improvement lies in the segmentation of the local models, which not only retains the benefits of contrastive learning for the shared layer models but also enhances the training process by incorporating the contrastive learning loss of the personalized layer models. As a result, the extracted information from each model becomes more diverse and comprehensive, thus leading to the observed improvement in the overall accuracy of the global model.

6. Conclusions

In this paper, we have proposed a new edge-assisted federated learning (FL) scheme in wireless network scenarios to address the issues of slow convergence speed and low global

model accuracy caused by Non-IID data in traditional FL architectures. We optimized the design in three aspects: update strategy, pairing method, and loss function. We adopted an edge–cloud asynchronous update approach and optimized the pairing of clients and servers in the asynchronous process, which significantly improved the effectiveness of the individual modules. Compared to the current traditional three-tier federated learning schemes, this method achieved a 2.5×–7.0× speedup in acceleration. Additionally, regarding the improvement on the loss function for local updates of the clients, we surpassed existing benchmarks on all three datasets by combining personalized ideas with the existing federated contrastive learning, thus achieving higher convergence speed.

Author Contributions: Conceptualization, W.S.; methodology, X.S. and Y.L.; validation, X.L.; writing—original draft preparation, X.S.; writing—review and editing, C.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the S&T Special Program of Huzhou under Grant 2022GZ14.

Data Availability Statement: Data is contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; Agüera y Arcas, B. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the Artificial Intelligence and Statistics, Ft. Lauderdale, FL, USA, 20–22 April 2017; PMLR: New York, NY, USA, 2017; pp. 1273–1282.
2. Nasr, M.; Shokri, R.; Houmansadr, A. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 19–23 May 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 739–753.
3. Kairouz, P.; McMahan, H.B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A.N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. Advances and open problems in federated learning. *Found. Trends Mach. Learn.* **2021**, *14*, 1–210. [CrossRef]
4. Lim, W.Y.B.; Luong, N.C.; Hoang, D.T.; Jiao, Y.; Liang, Y.C.; Yang, Q.; Niyato, D.T.; Miao, C. Federated Learning in Mobile Edge Networks: A Comprehensive Survey. *IEEE Commun. Surv. Tutor.* **2019**, *22*, 2031–2063. [CrossRef]
5. Liu, L.; Zhang, J.; Song, S.H.; Letaief, K.B. Client-Edge-Cloud Hierarchical Federated Learning. In Proceedings of the IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020; pp. 1–6.
6. Wu, Q.; Chen, X.; Ouyang, T.; Zhou, Z.; Zhang, X.; Yang, S.; Zhang, J. Hiflash: Communication-efficient hierarchical federated learning with adaptive staleness control and heterogeneity-aware client-edge association. *IEEE Trans. Parallel Distrib. Syst.* **2023**, *34*, 1560–1579. [CrossRef]
7. Xu, M.; Fu, Z.; Ma, X.; Zhang, L.; Li, Y.; Qian, F.; Wang, S.; Li, K.; Yang, J.; Liu, X. From cloud to edge: A first look at public edge platforms. In Proceedings of the 21st ACM Internet Measurement Conference, Virtual, 2–4 November 2021; pp. 37–53.
8. Tran, N.H.; Bao, W.; Zomaya, A.Y.; Nguyen, M.N.H.; Hong, C.S. Federated Learning over Wireless Networks: Optimization Model Design and Analysis. In Proceedings of the IEEE INFOCOM 2019—IEEE Conference on Computer Communications, Paris, France, 29 April–2 May 2019; pp. 1387–1395.
9. Chen, M.; Shlezinger, N.; Poor, H.V.; Eldar, Y.C.; Cui, S. Communication-efficient federated learning. *Proc. Natl. Acad. Sci. USA* **2021**, *118*, e2024789118. [CrossRef] [PubMed]
10. Jiang, M.; Wang, Z.; Dou, Q. HarmoFL: Harmonizing Local and Global Drifts in Federated Learning on Heterogeneous Medical Images. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtual, 2–9 February 2021; pp. 1087–1095.
11. Li, Q.; Diao, Y.; Chen, Q.; He, B. Federated Learning on Non-IID Data Silos: An Experimental Study. In Proceedings of the IEEE 38th International Conference on Data Engineering (ICDE), Kuala Lumpur, Malaysia, 9–12 May 2022; pp. 965–978.
12. Li, T.; Sahu, A.K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; Smith, V. Federated optimization in heterogeneous networks. *Proc. Mach. Learn. Syst.* **2020**, *2*, 429–450.
13. Karimireddy, S.P.; Kale, S.; Mohri, M.; Reddi, S.J.; Stich, S.U.; Suresh, A.T. SCAFFOLD: Stochastic Controlled Averaging for Federated Learning. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 5132–5143.
14. Li, Q.; He, B.; Song, D.X. Model-Contrastive Federated Learning. In Proceedings of the Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 10708–10717.
15. Ghosh, A.; Hong, J.; Yin, D.; Ramchandran, K. Robust federated learning in a heterogeneous environment. *arXiv* **2019**, arXiv:1906.06629.
16. Ghosh, A.; Chung, J.; Yin, D.; Ramchandran, K. An Efficient Framework for Clustered Federated Learning. *IEEE Trans. Inf. Theory* **2020**, *68*, 8076–8091. [CrossRef]
17. Gong, B.; Xing, T.; Liu, Z.; Xi, W.; Chen, X. Adaptive client clustering for efficient federated learning over non-iid and imbalanced data. *IEEE Trans. Big Data* **2022**, *99*, 1–15. [CrossRef]

18. Liu, B.; Guo, Y.; Chen, X. PFA: Privacy-preserving federated adaptation for effective model personalization. In Proceedings of the Web Conference 2021, Ljubljana, Slovenia, 19–23 April 2021; pp. 923–934.
19. Xia, W.; Wen, W.; Wong, K.; Quek, T.Q.S.; Zhang, J.; Zhu, H. Federated-Learning-Based Client Scheduling for Low-Latency Wireless Communications. *IEEE Wirel. Commun.* **2021**, *28*, 32–38. [CrossRef]
20. Dinh, T.Q.; Nguyen, D.N.; Hoang, D.T.; Pham, T.V.; Dutkiewicz, E. In-Network Computation for Large-Scale Federated Learning Over Wireless Edge Networks. *IEEE Trans. Mob. Comput.* **2021**, *22*, 5918–5932. [CrossRef]
21. Sattler, F.; Wiedemann, S.; Müller, K.R.; Samek, W. Robust and Communication-Efficient Federated Learning from Non-i.i.d. Data. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *31*, 3400–3413. [CrossRef] [PubMed]
22. Liu, S.; Yu, G.; Yin, R.; Yuan, J.; Shen, L.; Liu, C. Joint Model Pruning and Device Selection for Communication-Efficient Federated Edge Learning. *IEEE Trans. Commun.* **2022**, *70*, 231–244. [CrossRef]
23. Reisizadeh, A.; Mokhtari, A.; Hassani, H.; Jadbabaie, A.; Pedarsani, R. FedPAQ: A Communication-Efficient Federated Learning Method with Periodic Averaging and Quantization. In Proceedings of the International Conference on Artificial Intelligence and Statistics, Online, 26–28 August 2020; pp. 2021–2031.
24. Qu, Z.; Duan, R.; Chen, L.; Xu, J.; Lu, Z.; Liu, Y. Context-Aware Online Client Selection for Hierarchical Federated Learning. *IEEE Trans. Parallel Distrib. Syst.* **2021**, *33*, 4353–4367. [CrossRef]
25. Wolfrath, J.; Sreekumar, N.; Kumar, D.; Wang, Y.; Chandra, A. HACCS: Heterogeneity-Aware Clustered Client Selection for Accelerated Federated Learning. In Proceedings of the 2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS), Lyon, France, 30 May–3 June 2022; pp. 985–995.
26. Wang, K.; He, Q.; Chen, F.; Jin, H.; Yang, Y. FedEdge: Accelerating Edge-Assisted Federated Learning. In Proceedings of the ACM Web Conference 2023, Austin, TX, USA, 30 April–4 May 2023; Association for Computing Machinery: New York, NY, USA, 2023; pp. 2895–2904.
27. Wang, H.; Yurochkin, M.; Sun, Y.; Papailiopoulos, D.; Khazaeni, Y. Federated learning with matched averaging. *arXiv* **2020**, arXiv:2002.06440.
28. Hsu, T.M.H.; Qi, H.; Brown, M. Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification. *arXiv* **2019**, arXiv:1909.06335.
29. Collins, L.; Hassani, H.; Mokhtari, A.; Shakkottai, S. Exploiting Shared Representations for Personalized Federated Learning. In Proceedings of the International Conference on Machine Learning, Virtual, 18–24 July 2021; pp. 2089–2099.
30. Duan, Q.; Huang, J.; Hu, S.; Deng, R.; Lu, Z.; Yu, S. Combining federated learning and edge computing toward ubiquitous intelligence in 6G network: Challenges, recent advances, and future directions. *IEEE Commun. Surv. Tutor.* **2023**, *25*, 2892–2950. [CrossRef]
31. Dennis, D.K.; Li, T.; Smith, V. Heterogeneity for the win: One-shot federated clustering. In Proceedings of the International Conference on Machine Learning, Virtual, 18–24 July 2021; PMLR: New York, NY, USA, 2021; pp. 2611–2620.
32. Long, G.; Xie, M.; Shen, T.; Zhou, T.; Wang, X.; Jiang, J. Multi-center federated learning: Clients clustering for better personalization. *World Wide Web* **2023**, *26*, 481–500. [CrossRef]
33. Duan, M.; Liu, D.; Ji, X.; Liu, R.; Liang, L.; Chen, X.; Tan, Y. FedGroup: Efficient Federated Learning via Decomposed Similarity-Based Clustering. In Proceedings of the 2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), New York, NY, USA, 30 September–3 October 2021; pp. 228–237.
34. Deng, L. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Process. Mag.* **2012**, *29*, 141–142. [CrossRef]
35. Krizhevsky, A.; Hinton, G. Learning multiple layers of features from tiny images. *Handb. Syst. Autoimmune Dis.* **2009**, *1*, 1–60.
36. Xiao, H.; Rasul, K.; Vollgraf, R. Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms. *arXiv* **2017**, arXiv:1708.07747.
37. Yurochkin, M.; Agarwal, M.; Ghosh, S.; Greenewald, K.; Hoang, N.; Khazaeni, Y. Bayesian nonparametric federated learning of neural networks. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; PMLR: New York, NY, USA, 2019; pp. 7252–7261.