# GSFL: A Privacy-preserving Grouping-Split Federated Learning Approach in Resource-constrained Edge Computing Scenarios

QI LIU[*], Jiangsu Province Engineering Research Center of Advanced Computing and Intelligent Services, China and Nanjing University of Information Science and Technology, School of Software, China
ZHILU WANG[*], Nanjing University of Information Science and Technology, School of Computer Science, China
XIAOKANG ZHOU[†], Kansai University, Faculty of Business Data Science, Japan and RIKEN Center for Advanced Intelligence Project, Japan
YONGHONG ZHANG, Nanjing University of Information Science and Technology, School of Automation, China
XIAODONG LIU, Edinburgh Napier University, School of Computing, UK
HAIYANG LIN, Nanjing University of Information Science and Technology, School of Software, China

The advancement of mobile multimedia communications, 5G, and Internet of Things (IoT) has led to the widespread use of edge devices, including sensors, smartphones, and wearables. This has generated in a large amount of distributed data, leading to new prospects for deep learning. However, this data is confined within data silos and contains sensitive information, making it difficult to be processed in a centralized manner, particularly under stringent data privacy regulations. Federated learning (FL) offers a solution by enabling collaborative learning while ensuring privacy. Nonetheless, data and device heterogeneity complicate FL implementation. This research presents a specialized FL algorithm for heterogeneous edge computing. It integrates a lightweight grouping strategy for homogeneous devices, a scheduling algorithm within groups, and a Split Learning (SL) approach. These contributions enhance model accuracy and training speed, alleviate the burden on resource-constrained devices, and strengthen privacy. Experimental results demonstrate that the GSFL outperforms FedAvg and SplitFed by 6.53× and 1.18×. Under experimental conditions with $\alpha = 0.05$, representing a highly heterogeneous data distribution typical of extreme Non-IID scenarios, GSFL showed better accuracy compared to FedAvg by 10.64%, HACCS by 4.53%, and Cluster-HSFL by 1.16%. GSFL effectively balances privacy protection and computational efficiency for real-world applications in mobile multimedia communications.

[*]Qi Liu and Zhilu Wang are co-first authors of the article.
[†]Corresponding author: zhou@kansai-u.ac.jp (Xiaokang Zhou)

Authors' addresses: Qi Liu, qi.liu@nuist.edu.cn, Jiangsu Province Engineering Research Center of Advanced Computing and Intelligent Services, Nanjing, China and Nanjing University of Information Science and Technology, School of Software, Nanjing, China; Zhilu Wang, zhiluwang@nuist.edu.cn, Nanjing University of Information Science and Technology, School of Computer Science, Nanjing, China; Xiaokang Zhou, zhou@kansai-u.ac.jp, Kansai University, Faculty of Business Data Science, Osaka, Japan and RIKEN Center for Advanced Intelligence Project, Tokyo, Japan; Yonghong Zhang, zyh@nuist.edu.cn, Nanjing University of Information Science and Technology, School of Automation, Nanjing, China; Xiaodong Liu, x.liu@napier.ac.uk, Edinburgh Napier University, School of Computing, Edinburgh, UK; Haiyang Lin, 202212490371@nuist.edu.cn, Nanjing University of Information Science and Technology, School of Software, Nanjing, China.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM 1556-4665/2025/3-ART
https://doi.org/XXXXXXX.XXXXXXX

CCS Concepts: • **Computing methodologies** → *Cooperation and coordination*; • **Security and privacy** → *Privacy protections*; • **Human-centered computing** → *Ubiquitous and mobile computing*.

Additional Key Words and Phrases: Federated learning, Split learning, Non-IID, Clustering

**ACM Reference Format:**
Qi Liu, Zhilu Wang, Xiaokang Zhou, Yonghong Zhang, Xiaodong Liu, and Haiyang Lin. 2025. GSFL: A Privacy-preserving Grouping-Split Federated Learning Approach in Resource-constrained Edge Computing Scenarios. *ACM Trans. Autonom. Adapt. Syst.* 1, 1 (March 2025), 30 pages. https://doi.org/XXXXXXX.XXXXXXX

## 1 INTRODUCTION

With the unceasing advancement of 5G technology and mobile multimedia communication, the ubiquitous deployment of Internet of Things (IoT) [9] sensors, wearable devices, and smartphones has led to a dramatic increase in the volume of private data emanating from a plethora of distributed nodes within edge networks. This surge provides substantial opportunities for deep learning applications, notably within edge-based autonomous systems, where mobile multimedia communication becomes a critical component. Such communication includes real-time video streaming, image recognition, and voice interactions, which are indispensable for the functioning of deep learning algorithms across the domains of computer vision [44], anomaly detection, and data mining [25]. In the current digital era, the abundance of data presents unparalleled opportunities for the advancement of deep learning, given that these algorithms necessitate copious volumes of data to construct high-performance models via iterative training processes. Various organizations are harnessing big data to optimize the processes and performance of Artificial Intelligence (AI) applications [28, 51].

However, the majority of this data is typically dispersed, existing in isolation, and forming distinct data silos. Relying solely on data from individual devices could lead to models trained with inherent biases due to the lack of comprehensive data. Moreover, a significant proportion of the data is inherently sensitive. As a result of competition and privacy concerns within industries, devices are unable to directly transmit private data to centralized servers for model training. Within edge-based autonomous systems, particularly those related to mobile multimedia communication, such as metaverse [58], deep vision [27], remote sensing systems [26], mobile edge networks [50], Internet of Vehicles (IoV) [11], and intelligent transportation systems (ITS) [55]. There is a growing need for solutions that ensure data privacy while simultaneously enabling efficient and autonomous communication. These systems elevate the requirements for low-latency, high-reliability, and privacy-protected multimedia data transmission.

With the continuous promulgation and enhancement of data privacy laws, such as the Cybersecurity Law of the People's Republic of China implemented on June 1, 2017, the Personal Information Protection Law that took effect on November 1, 2021, and the General Data Protection Regulation (GDPR) [42] introduced by the European Union in 2018. In response, society is placing greater importance on safeguarding information security and ensuring the protection of personal privacy. Companies and institutions are intensifying their commitment to data security and the protection of user privacy. Traditional distributed collaborative learning methods have not adequately addressed data privacy issues [23], as they involve sending users' private data to central servers for model training, leading directly to the leakage and misuse of personal privacy information and data. These enacted regulations pose novel challenges to the distributed data processing practices in the AI field, with a rapidly growing demand for distributed AI solutions that safeguard privacy.

In recent years, researchers have been focusing on the issue of enabling data to be used in distributed collaborative learning without compromising data privacy. Specifically, they are exploring methods to collaboratively train deep learning models with multiple parties without the need to upload local user data. In 2016, McMahan et al. [31] at Google first introduced the concept of

Federated Learning (FL), a novel paradigm for distributed collaborative learning. The prevalent FL framework presumes that data holders (clients) form a federation, with edge devices serving as clients, collectively training the FL model under the coordination of a central parameter server (the FL server). FL allows numerous clients to conduct model collaborative training among multiple data silos (or devices) without uploading local private data by transmitting model parameters or other intermediate results. During the FL process, clients utilize their private data to train local models. These trained parameters are then sent to the central server. The central server aggregates the global model by combining the models received from all clients in each round of FL communication. Subsequently, the central server broadcasts the global model to all clients for iterative updating. It is evident that FL realizes the maximum utilization of data value on the basis of ensuring data privacy, especially in the aspect of the Internet of Medical Things (IoMT). [54]

However, real-world scenarios pose significant challenges of heterogeneity [47] and resource constraints [24], particularly within large-scale and complex edge networks. Heterogeneity encompasses data distribution heterogeneity among edge devices, known as data heterogeneity, and variations in hardware capabilities of devices, known as device heterogeneity. The challenge of resource constraints pertains to the limitations in computational capability, storage capacity, and energy consumption of edge devices. Devices such as smartphones, sensors, and cameras are commonly equipped with limited processor speeds, small-scale memory, and restricted battery capacity. These limitations mean that such devices are incapable of performing complex computational tasks or operating continuously without frequent recharging. Resource scheduling [10] and task scheduling [21] are critically important in practical distributed computing environments. Therefore, the design of algorithms for edge computing must account for the characteristics of these resource-constrained devices, aiming to minimize computation and communication loads without sacrificing performance.

Since data is independently generated based on the behavior of terminal devices, its distribution is typically Non-Independent and Identically Distributed (Non-IID). In the case of data generation by terminal devices, if the user behavior patterns of different terminal devices vary significantly, then the data generated by each terminal device will have unique characteristics and distributions, which leads to the Non-IID of the data. Consequently, local datasets may not fully represent the overall distribution, potentially leading to biased model updates and diminished model accuracy. In application scenarios related to mobile multimedia communications, data often also exhibits Non-IID characteristics.

In the face of data heterogeneity, common FL approaches use a single global model that is shared by all clients for training across the whole federation. However, a single global model cannot adequately accommodate all the diverse local data distributions. Google's original FL algorithm, FedAvg, exhibits poor convergence with highly heterogeneous data, manifesting in a significant reduction in the accuracy of the FL model when learning from Non-IID data [30]. The decline in model performance is attributed to the client drift phenomenon [20], where local models, starting from the same initial values, converge to different models after several rounds of local training and global aggregation. This introduces unknown biases into the global model during the aggregation phase, reducing the ultimate precision of the FL model and hindering convergence, thereby extending the duration of FL training. In real-time application scenarios of mobile multimedia communications, quickly training an effective model is of crucial importance for business development. If the model training efficiency is low and it cannot adapt to heterogeneous data in a timely manner, it will be unable to provide strong support for services, thus seriously affecting the operating efficiency of the entire system.

In practical applications such as health monitoring with medical IoT devices [59], privacy protection in mobile robotic systems [56], collaborative learning in automated vehicles operating over high-speed mobile networks [60], and precise positioning within intelligent transportation systems [57], FL networks may involve a substantial number of various devices involved in mobile multimedia communications, such as smartphones, sensors, and cameras. These IoT devices have obvious differences in hardware capabilities. The disparity in hardware capabilities inevitably leads to device heterogeneity [33], resulting in different storage capacity and computational capability. For instance, in a mobile edge network, one client may be a smartphone while another is a smartwatch. Compared to a smartwatch, a smartphone has greater storage capacity and stronger computational power, which introduces device heterogeneity. Consequently, within the FL training process, devices such as smartwatches may be subject to extended local runtimes. This prolongation increases the likelihood of these devices being inadvertently discarded or lost. Such occurrences can detract from the efficiency and stability of the FL system. In FL, clients are tasked with performing local updates and transmitting the updated weight parameters back to the server side. However, client devices might encounter failures during this process. During the synchronization phase for model updates, devices characterized by limited computational resources may necessitate considerable time to implement the model parameter update, potentially leading to their designation as stragglers within the FL network. In general, device heterogeneity leads to system imbalances and inefficiencies due to varying computational speeds or resources among different clients, causing system delays or bottlenecks. Furthermore, the aforementioned scenario contributes to uncertainty and instability within the system, stemming from the diverse states of client devices, which may range from optimal functioning to system failures or complete losses. Moreover, device heterogeneity will also lead to uneven security and privacy protection capabilities of different devices in processing data and participating in the FL process. Some devices, due to resource constraints, cannot adopt complex privacy protection technologies. Consequently, the heterogeneity of devices presents challenges that necessitate addressing issues such as straggler mitigation [36] and ensuring fault tolerance within the FL framework. This requires the implementation of adaptive mechanisms capable of managing the variable feedback from a range of devices in large-scale resource-constrained edge computing scenarios.

To augment the efficacy of FL within heterogeneous and resource-constrained scenarios, scholars have introduced a multitude of strategies aimed at bolstering model precision despite the challenges posed by data heterogeneity [40]. These include methods based on similarity, such as Multi-Task Learning (MTL) [2, 16, 38], model interpolation [5, 6, 13], and clustering [1, 7, 12, 22, 29, 35, 43], which aim to mitigate the issue of weight divergence caused by client drift phenomena. However, most methods addressing Non-IID data have significant side effects due to substantial computational and communication overhead. Moreover, some approaches expose clients' raw data distribution features to the server side for comparison, potentially leading to the leakage of user privacy data. In response to device heterogeneity, academic circles have also proposed combining FL with Split Learning (SL) mechanisms [4, 14, 15, 18, 37, 41, 46, 48, 52, 53]. In scenarios with limited computational and storage resources, these mechanisms enhance training efficiency and reduce device load. Furthermore, as the model architecture is partitioned between the client and the server, SL offers better model privacy than FL, and the integration of SL mechanisms significantly strengthens privacy protection.

Contemporary leading-edge methods for bolstering data privacy and security within the scope of FL incorporate an array of sophisticated techniques. These include Differential Privacy (DP) [49], secure multi-party computation (MPC) [8], and homomorphic encryption [32]. Each of these approaches contributes uniquely to the overarching objective of preserving the confidentiality and integrity of data during the FL process. Research by Song et al. [39] has shown that FL integrated with these privacy protection technologies can provide robust security. Nevertheless, these methods still face issues, as encryption technologies like homomorphic encryption and DP require extensive computational resources, and the key generation process during encryption is overly complex. The performance drops noticeably as the number of clients increases. Furthermore, the use of DP will reduce data utility, and the computational and communication costs of homomorphic encryption and MPC are quite high [17]. Challenges related to computational overhead and privacy protection hinder the widespread deployment of FL in heterogeneous and resource-constrained edge computing scenarios. Designing accurate and efficient FL methods that do not expose user privacy information has become one of the main bottlenecks in the design of heterogeneous IoT edge intelligent systems. By optimizing existing FL architectures and training methods to alleviate these issues, enhancing the adaptability of FL algorithms to data heterogeneity and device heterogeneity, improving model accuracy and training efficiency, while protecting personal data privacy and reducing the risk of data misuse, it can achieve practical and significant real-world applications. Through these optimizations, this work will provide a safer and more efficient FL method for mobile multimedia communication in mobile edge computing within heterogeneous and resource-constrained scenarios.

In the context of mobile multimedia communications, the method proposed in this paper has multiple advantages to address related challenges. Firstly, for the resource-constrained edge computing scenario, in order to ensure low latency, lightweight grouping strategies, intra-group selection strategies, as well as model decomposition and adaptive channel pruning measures are adopted. By grouping through lightweight hash encryption functions and selecting clients through the Contextual Multi-armed Bandit (CMAB) strategy, the delay caused by excessive participation of resource-constrained devices in training is avoided. At the same time, model decomposition and channel pruning reduce the computing burden of devices, improve processing speed, and reduce latency. Secondly, in terms of privacy protection, lightweight grouping strategies and split federated learning mechanisms are employed. These measures meet the requirements of privacy protection by reducing the transmission of original data and limiting the scope of data interaction, minimizing the exposure of privacy data and controlling data access. Furthermore, the proposed algorithm can be adaptively adjusted. At the data level, it adapts to data changes through data feature extraction and update as well as grouping based on data distribution. At the device level, it copes with device changes by using device capability perception and client scheduling as well as dynamic adaptation of model decomposition and channel pruning. Finally, in the complex environment of edge-based autonomous systems, the GSFL algorithm guarantees the stability and reliability of mobile multimedia communications. The grouping strategy has fault tolerance. Grouping based on data similarity enables devices in the same group to continue interacting in case of failures, and the intra-group selection strategy can dynamically adjust client selection. The SL mechanism is robust. Model decomposition allows the device-side and server-side models to continue operating in case of failures, and adaptive channel pruning makes the model more stable in resource-constrained scenarios.

The main contributions of this paper are as follows:

- **Innovative FL Approach**: This paper introduces GSFL, a novel FL algorithm that innovatively combines a lightweight, dual-layer client selection method with a SL mechanism. Tailored for heterogeneous and resource-constrained edge computing scenarios, GSFL enhances privacy and reduces computational overhead. The first layer of this approach groups devices based on data similarity using a lightweight hash encryption function, while the second layer optimizes training efficiency by selectively activating clients within these groups based on their resource capabilities. Unlike certain methods that potentially entail the risk of privacy leakage, GSFL better protects users' data privacy through grouping and SL mechanisms. This strategic integration of grouping and SL mechanisms ensures both efficient resource utilization and improved data privacy.

- **Performance under Non-IID Conditions**: The dual-layer client selection method of GSFL has been optimized for the Non-IID problem caused by different degrees of data heterogeneity. Firstly, lightweight grouping is performed according to the data similarity of devices, reducing unnecessary computing and storage overhead and enabling resources to be more reasonably allocated to key tasks on devices. Then, the CMAB strategy is used to schedule clients within groups for training, and training tasks are reasonably arranged according to the resource status of devices to avoid excessive consumption of resources. This unique design enables GSFL to better capture data features and reduce model update deviation when dealing with Non-IID data. Thus, in comparison with traditional FL methods such as FedAvg and other advanced strategies such as HACCS, SplitFed, and Cluster-HSFL, it not only achieves higher accuracy but also significantly improves training efficiency.
- **Significantly Reducing Computational Load**: The SL mechanism adopted by GSFL effectively reduces the computational load on the device side through model decomposition and adaptive channel pruning. In the process of model decomposition, an appropriate splitting point is selected according to the computing power of the device to divide the model into two parts, the device side and the server side, so that the device only needs to process the model part suitable for its own resources. At the same time, the adaptive channel pruning strategy removes unimportant channels and reduces parameters and computational complexity. This approach avoids the problem that the device cannot efficiently run the model due to limited computing resources. Compared with methods that rely on complex encryption technologies and have high computational overhead, it significantly improves the operating efficiency of the device in resource-constrained scenarios.
- **Adaptive adjustment and stability guarantee**: The proposed GSFL algorithm shows strong adaptability and effective guarantee for the stability and reliability of mobile multimedia communications. In terms of adaptive adjustment, at the data level, data feature extraction and grouping based on data distribution are used to adapt to data changes. At the device level, device capability perception and client scheduling as well as dynamic adaptation of model decomposition and channel pruning are used to deal with device changes. In terms of stability and reliability guarantee, the fault tolerance of the grouping strategy enables devices in the same group to continue interacting in case of failures and the client selection within the group can be dynamically adjusted. The robustness of the SL mechanism allows the device-side and server-side models to continue operating in case of failures. Adaptive channel pruning makes the model more stable in resource-constrained scenarios, thus ensuring the stable and reliable operation of mobile multimedia communications.

## 2 RELATED WORK

In the realm of FL, data heterogeneity, device heterogeneity, and privacy protection have always been focal points of research. Particularly in resource-constrained edge computing scenarios, the limitations of devices' computational capability, storage capacity, and energy consumption impose heightened demands on the performance and efficiency of FL systems. Accordingly, researchers have explored a variety of optimization strategies to accommodate these heterogeneous, resource-constrained edge devices. These strategies aim to resolve the inconsistency in client data distributions, alleviate the limitations of computational capability and storage capacity of devices, and strengthen privacy protection. This will ultimately improve the feasibility and effectiveness of FL in real-world applications.

## 2.1 Similarity-based federated learning algorithm

Recent works have been dedicated to addressing performance issues caused by client drift when learning on Non-IID data in FL. In contrast to the conventional approach of training a singular global FL model, these strategies cultivate personalized models through the modification of the FL model's aggregation process. A significant amount of research has been conducted on personalized FL methods based on similarity, with MTL and model interpolation focusing on pairwise relationships between clients, while clustering looks at group relationships among clients.

*2.1.1 Multi-task learning.* MTL aims to train models that are proficient in concurrently executing multiple related tasks by exploiting shared domain-specific knowledge across these tasks to enhance generalization capabilities. In the context of FL, this involves regarding each client as an individual task within an MTL framework, thereby enabling the central server to discern and exploit potential correlations among clients, as reflected by their heterogeneous local datasets. Smith et al. [38] expanded distributed MTL into the domain of FL by devising a method for the building of personalized models tailored to individual clients. However, their approach necessitates the involvement of all participants in each training iteration, a requirement that may impede scalability and practicality. The proposed MOCHA framework, while pioneering, is confined to convex models, which limits its extensibility and generalization capabilities within the realm of deep learning applications. Addressing these limitations, Corinzia et al. [2] introduced the VIRTUAL algorithm, a virtual federated MTL method employing Bayesian techniques for variational inference, capable of accommodating non-convex models. Despite its theoretical advancements, VIRTUAL is computationally intensive, posing challenges for large-scale IoT edge FL implementations. In an effort to refine model personalization within FL, Huang et al. [16] proposed the federated attentive message passing (FedAMP) approach. FedAMP utilizes an attention mechanism to promote pairwise collaboration among clients with similar data distributions, supported by personalized cloud models for each client. Nonetheless, this approach imposes significant computational and communicative burdens on the server, particularly within the scope of large-scale edge FL scenarios. Hanzely and Richtárik [13] introduced an innovative method that integrates global and local model components to craft personalized FL models. This method strives to strike a balance between generalization and personalization across the server-hosted global models and the client-specific local models.

*2.1.2 Model interpolation.* Deng et al. [5] presented the adaptive personalized FL (APFL) algorithm, which incorporates client-specific parameters to dynamically adjust the impact of local and global models during the FL training process. This adaptive weighting aims to optimize personalization for each individual client. Diao et al. [6] introduced HeteroFL, an innovative FL framework that adeptly constructs adaptive personalized local models for each client by utilizing a centralized global model. This method is designed to effectively mitigate performance challenges arising from data heterogeneity. Model interpolation techniques employed within this framework utilize the global model as a foundation for personalization, necessitating a meticulous determination of the optimal synthesis of local and global models, referred to as the most favorable interpolation. This complex interaction between the clients and the server during the model interpolation process induces additional computational and communicative burdens. Such overheads have the potential to inflate operational costs, a consideration that is particularly pertinent in large-scale FL scenarios with inherent resource limitations. Furthermore, the sharing of local information, which is crucial for the implementation of model interpolation, has the potential to jeopardize client confidentiality, thus adding a substantial risk that needs to be addressed.

*2.1.3 Clustering.* For scenarios with significantly different clients or data distributions, training a global FL model using a client-server architecture is not the optimal choice. Grouping the clients and then training a FL model for each group of clients, resulting in a multi-model approach, is more suitable for resource-constrained and data-heterogeneous scenarios. Liu et al. [29] introduced an edge-based FL method, termed eFL, which clusters end devices using model parameters. This method calculates the cosine similarity across multiple dimensions between local and global model parameters to assess the need for local updates, thereby reducing superfluous communication. Devices are grouped based on network proximity and interact with the cloud through mobile edge nodes within their clusters. In the clustered FL (CFL) algorithm, Sattler et al. [35] employed an optimal bisection algorithm predicated on the cosine similarity of client gradient updates to categorize FL clients into distinct clusters. However, the need for multiple communication rounds to separate all inconsistent clients renders this method computationally and communicatively demanding, potentially undermining its feasibility in resource-constrained scenarios. Briggs et al. [1] devised a FL with hierarchical clustering (FL+HC) that initially trains a global model for a predetermined number of iterations before employing a one-shot clustering of all clients based on their gradient updates. Subsequent FL training is conducted independently within each cluster, producing several FL models. While this approach is suitable for Non-IID data distributions, it necessitates additional infrastructure to facilitate the hierarchical clustering and demands significant computational resources to determine pairwise distances among clients. Ghosh et al. [12] proposed the iterative federated clustering algorithm (IFCA), which eschews the creation of a single global model in favor of multiple models, each serving different client clusters. During the model aggregation phase, the server consolidates models within each cluster. Duan et al. [7] introduced FedGroup, a FL architecture that employs a static clustering strategy for clients and incorporates a cold-start process for integrating new clients. The architecture utilizes Euclidean distance based on decomposed cosine similarity and adopts the k-means++ algorithm to cluster local client updates, thereby enhancing the efficiency of the FL process.

Although these clustering algorithms are innovative, they indeed share some common challenges. For instance, both IFCA and FedGroup require a predetermined number of clusters, which is not scalable for large numbers of edge devices. Moreover, eFL, CFL, FL+HC, IFCA, and FedGroup only compare the Euclidean distance or cosine distance between different clients' neural network update parameters, which may lead to poor model similarity results and weak mitigation of Non-IID effects. Liu et al. [22] proposed PFA, which groups clients with similar data distributions to collaborate, ultimately providing clients with personalized FL models. However, PFA does not offer formal privacy guarantees when extracting data distribution features, and it uses the FedAvg algorithm to aggregate intra-group clients during group training without considering resource-constrained edge scenarios, potentially leading to stragglers affecting the efficiency of FL. Wolfrath et al. [43] proposed HACCS, a heterogeneity-aware clustering client selection method that extracts device data distribution histograms and uses DP techniques to protect user privacy before clustering by the server. However, applying DP directly to data distribution features reduces data utility, leading to suboptimal clustering results and, consequently, reduced accuracy of the FL models obtained from clustered FL.

## 2.2 Split-based federated learning algorithm

In scenarios with heterogeneous devices, SL has been incorporated into FL to enhance training efficiency and reduce client-side burden, especially in scenarios constrained by computational and storage resources. Additionally, the SL mechanism partitions the FL model architecture between the client and server, with neither side having access to the models of the other. The exclusive use of either client or server models makes reconstruction attacks challenging, thus enhancing the privacy of users' local data and strengthening the privacy protection capabilities of FL.

He et al. [14] developed FedGKT, a framework that integrates SL with knowledge distillation, utilizing two types of models. This structure integrates a computationally lightweight feature extractor deployed on local devices, thereby diminishing the device's computational demands, and a more sophisticated model resides on the server. This configuration facilitates bidirectional knowledge transfer: the server acquires feature maps and corresponding soft labels from the devices, while the devices, in turn, are provisioned with soft labels from the server. The resultant comprehensive model emerges as a synthesis of the insights gleaned from both the device and server models, with the knowledge exchange being orchestrated asynchronously to mitigate the impact of straggler effects. This innovative approach enhances the overall performance and efficiency of the FL process. Thapa et al. [41] proposed the SplitFed algorithm, which addresses the inherent limitations of slower execution speed in SL compared to FL and weaker privacy protection in SL compared to FL. This algorithm combines the advantages of SL and FL to overcome these drawbacks. The model is split and trained on both clients and servers, significantly diminishing the computational demands on clients, making it suitable for resource-constrained devices, while integrating DP to strengthen data privacy. Shen et al. [37] developed the RingSFL algorithm that addresses client heterogeneity issues through a ring topology and model partitioning mechanism. It reduces training time by adaptively allocating computational load and employs split models to lower the resource demands on clients. This significantly reduces the load on resource-constrained devices and enhances data privacy through model partitioning.

Zhang et al. [52] proposed the Cluster-HSFL algorithm, which combines clustering, SL, and FL to amplify model training parallelism and reduce clients' computational burdens. In addition to elevating privacy protection, model partitioning diminishes the workload on individual clients, thereby lowering training latency. Zheng et al. [53] introduced the PPSFL algorithm, which merges FL with SL and proposes a novel strategy for model decomposition, mitigating the risk of information leakage from intermediate results. The integration of Group Normalization (GN) layers into the network, with private GN layers, alleviates the negative impact of heterogeneity on training while promoting personalization for each client model. Khan et al. [18] put forward HSFL (Hierarchical Split FL), a new framework designed to address the issues of resource limitation, single-point failures, and slow convergence encountered by IoT devices during FL. This framework takes into account the different capabilities of heterogeneous devices and reduces the workload on clients through SL. Deng et al. [4] introduced an innovative hybrid framework that integrates SL with FL. This framework adaptively partitions the machine learning model into sub-models designated for the client and server sides, respectively, with the aim of delivering intelligent services within the context of resource-constrained IoT scenarios. This research developed a lightweight, contextual bandit learning-based adaptive model decomposition approach, offloading part of the model training tasks to edge servers and experimentally validating its effectiveness in reducing training latency and protecting data privacy.

He et al. [15] explored the domain of Unmanned Aerial Vehicles (UAVs), presenting a novel hybrid method that combines SL with Multi-Agent Reinforcement Learning (MARL) within a FL paradigm, denoted as SFL-MARL. This innovative approach is tailored to augment the efficacy of UAV swarms operating in 5G networks, with a dual focus on minimizing communication overhead and upholding privacy. The SL component of the approach effectively harnesses the computational capability inherent to each UAV, thereby contributing to the refinement of model precision. Through this approach, UAVs are empowered to conduct local machine learning model training and to securely transmit encrypted model parameters to a centralized server utilizing federated averaging techniques. Following the aggregation and enhancement of the model by the server, the refined model is redistributed to each UAV, enabling local adjustments and fine-tuning. This strategic integration of SL and MARL within a FL framework presents a promising direction for optimizing UAV swarm operations in next-generation wireless networks. Yin et al. [48] introduced a novel hybrid federated SL framework that utilizes the collaborative training of multiple working nodes in FL and the low computational load characteristic of SL to optimize model training in wireless networks. To mitigate the computational idleness resulting from model partitioning, the researchers designed a parallel computation scheme without sharing labels and conducted a theoretical analysis of the impact of delayed gradients on convergence. Moreover, they employed Generative Adversarial Networks (GANs) and multi-objective optimization to enhance predictive performance. Yang et al. [46] focused their research on medical imaging networks, incorporating homomorphic encryption into the training process to bolster privacy protection and alleviate client-side load. Their approach facilitated the training of U-Net architectures while enhancing privacy safeguards and reducing data volume requirements per client through SL.

The aforementioned studies indicate a significant volume of research on FL in heterogeneous and resource-constrained edge computing. FL algorithms based on similarity particularly address performance issues caused by client drift stemming from data heterogeneity, while those based on SL focus on overcoming resource bottlenecks and straggler problems due to device heterogeneity. Existing similarity-based FL algorithms, such as MTL and model interpolation, have to contend with the high computational and communication costs associated with pairwise client relationships. Moreover, model interpolation requires sharing of local information, which may expose clients to privacy risks. When model parameters are used for clustering, the results may not be optimal, and the mitigation effect for Non-IID data can be weak. In contrast, using data distribution characteristics for clustering can reduce data utility due to DP, thereby compromising the accuracy of the FL model.

Algorithms based on SL significantly enhance privacy protection, but most methods have only addressed challenges arising from device heterogeneity, without considering the setting under Non-IID data heterogeneity. The SL mechanism introduces additional computational and communication overheads, and certain methods lack essential lightweight techniques. Furthermore, the introduction of heavyweight technologies such as MARL and homomorphic encryption incurs substantial additional computational costs. Therefore, there is an urgent need to optimize existing FL methods to comprehensively consider the challenges of heterogeneity, resource limitations, and privacy constraints. The goal is to improve the adaptability of FL algorithms to data and device heterogeneity, enhance model accuracy and training efficiency, and simultaneously reduce the risk of data misuse and protect personal data privacy.

Differing from existing methods, this paper innovatively combines a lightweight, dual-layer client selection method with a FL mechanism. In the first layer of the dual-layer client selection method, lightweight hash encryption function is utilized to group devices based on data similarity. The second layer selectively activates clients within these groups based on their resource capabilities, thereby optimizing training efficiency. Subsequently, split and parameter aggregation are conducted using a federated server, while training is performed using a cloud server. The proposed GSFL algorithm is specifically tailored for heterogeneous and resource-constrained edge computing scenarios, enhancing privacy and reducing computational overhead. This strategic integration of grouping and SL mechanisms ensures efficient resource utilization and improved data privacy.

## 3 METHODOLOGY

### 3.1 Lightweight grouping federated learning method based on dual-layer client selection

Given the vast amount of data samples acquired in edge computing and the limited computational resources of IoT devices for extracting data distribution features, directly collecting and transmitting

Table 1. Main notation in GSFL.

| Notation | Meaning |
|---|---|
| $C$ | The collection of devices, denoted as $C = \{c_1, c_2, \ldots, c_n\}$, the quantity of devices is denoted as $n$. Here, $c_i$ represents the $i^{th}$ device |
| $d_i$ | The device $c_i$ holds a private dataset, where $\|d_i\|$ represents the size of the dataset |
| $x_{i,j}$ | The $j^{th}$ data sample on device $c_i$ |
| $y_{i,j}$ | The corresponding label of the $j^{th}$ data sample on device $c_i$ |
| $K$ | The number of groups |
| $G$ | The grouping result set of homogeneous devices |
| $P$ | Local model training rounds |
| $v_i$ | The vectorization result of extracting the ReLU layer feature map during pre-training FL on device $c_i$ |
| $h_i$ | The hash value obtained using a lightweight hash encryption function |
| $H$ | The hash value set of encrypted data distribution features $h_i$ |
| $M$ | The similarity matrix used to compare the similarity of data distributions |
| $E$ | Intra-group model training rounds |
| $V$ | Selected device set |
| $W$ | The weight parameters of the FL model |

these features could increase the risk of privacy breaches. This reality is in conflict with the primary goal of FL to enhance privacy protection within distributed machine learning. The overall framework of the lightweight grouping FL method based on dual-layer client selection is illustrated in Fig. 1. The first layer of client selection involves client grouping, which is collaboratively conducted by the data feature extraction module (DFE) and the homogeneous device grouping module (HDG). The second layer of client selection is carried out by the intra-group client scheduling module (ICS). To achieve efficient personalized FL in a data-heterogeneous, resource-constrained edge network, a data feature extraction module is essential. In the proposed similarity-based grouping FL framework, acquiring data distribution features is a prerequisite for device grouping.

Considering the high heterogeneity of edge device data, the challenges of Non-IID data on the speed and convergence of FL in real-world situations, and the limited network connectivity of IoT devices in resource-constrained edge scenarios, it is impractical to require FL to perform model updates and aggregation in parallel across all participating devices. In light of this, a homogenous device grouping module was devised, primarily responsible for grouping similar devices at the edge server, using similarity-based grouping methods to mitigate the impact of Non-IID data on the performance of FL models in highly heterogeneous data scenarios. In heterogeneous data scenarios, the edge server categorizes devices into homogenous groups and coordinates the FL process within each group. Each device shares gradient updates only within its group, resulting in distinct global models for different groups. Given that improper scheduling may lead to decreased learning efficiency and delay issues, particularly in resource-constrained edge settings, the accuracy of FL training depends on capturing the unique data distributions rather than necessarily involving all clients. Therefore, based on device grouping, a certain number of devices are selected within each homogenous group for FL training. Table 1 summarizes the main notation used in this article.

### 3.1.1 *Data feature extraction module.*

Before device grouping and grouping FL start, the data feature extraction module preprocesses the data distribution of the devices. The edge server broadcasts the data feature extraction model. Each edge device $c_i$ performs pre-training on local private data $d_i$

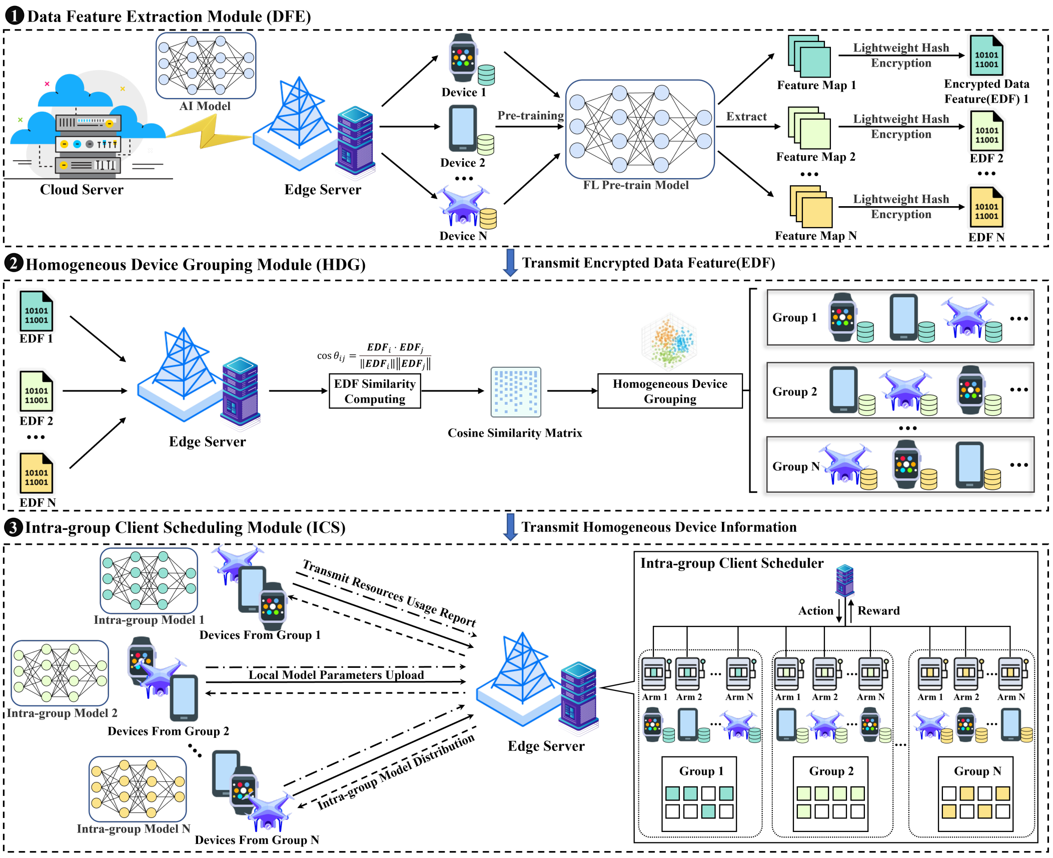

❶ Data Feature Extraction Module (DFE)
AI Model
Cloud Server
Edge Server
Device 1
Device 2
Device N
Pre-training
FL Pre-train Model
Extract
Feature Map 1
Feature Map 2
Feature Map N
Lightweight Hash Encryption
Encrypted Data Feature(EDF) 1
EDF 2
EDF N
10101 11001
Transmit Encrypted Data Feature(EDF)
❷ Homogeneous Device Grouping Module (HDG)
EDF 1
EDF 2
EDF N
Edge Server
$\cos\theta_{ij} = \frac{EDF_i \cdot EDF_j}{\|EDF_i\|\|EDF_j\|}$
EDF Similarity Computing
Cosine Similarity Matrix
Homogeneous Device Grouping
Group 1
Group 2
Group N
Transmit Homogeneous Device Information
❸ Intra-group Client Scheduling Module (ICS)
Intra-group Model 1
Devices From Group 1
Intra-group Model 2
Devices From Group 2
Intra-group Model N
Devices From Group N
Transmit Resources Usage Report
Local Model Parameters Upload
Intra-group Model Distribution
Edge Server
Intra-group Client Scheduler
Action
Reward
Arm 1
Arm 2
Arm N
Group 1
Group 2
Group N

Fig. 1. The overall framework diagram of lightweight grouping FL method based on dual-layer client selection.

before grouping. During the pre-training phase, the feature map is extracted from the Rectified Linear Unit (ReLU) layer of the Convolutional Neural Network (CNN) and uses a lightweight hash function encrypts the ReLU layer feature map to generate the encrypted data feature $h_i$ (ie, the encrypted data feature EDF), and then uploads it to the server for the next step of device grouping. When the data situation changes, for example, due to changes in user usage habits or the influence of environmental factors, the data features collected by devices may change. At this time, devices can be pre-trained again, extract new data features and encrypt them before uploading to the server. The server can regroup according to the new data features to adapt to the changes in data.

The data feature extraction module centers around employing the ReLU layer feature maps of all devices by means of a lightweight hash encryption function for the generation of EDF. The lightweight hash encryption function PHOTON-Beetle-Hash [19] used is mainly designed for resource-constrained devices, enhancing user privacy protection for various devices. The function's small code size and low RAM requirements, coupled with low computational complexity, minimize additional computational costs, making it ideal for resource-constrained edge computing systems. After dimensionality reduction, these data features also reduce the communication overhead between the device and the server. In mobile multimedia communication scenarios with a large number of IoT devices participating, the delay caused by directly transmitting a substantial number of original data features is avoided, thereby improving the efficiency of FL.

*3.1.2 Homogeneous device grouping module.* The proposed homogeneous device grouping module performs a second stage of preprocessing on the devices before formally performing grouping FL. First, the edge server collects the EDF sent by all devices in the domain, that is, the encrypted data feature $h_i$, and stores it in the set $H$. Then, homogeneous device grouping module uses cosine similarity to measure the similarity of data distribution between different devices, and saves the calculation result of cosine similarity in a metric matrix S that can represent the degree of similarity of data distribution between clients, such as shown in formula (1), $\mathrm{S}_{ij}$ is the cosine similarity between the two encrypted data features $h_i$ and $h_j$, that is, the data distribution similarity between devices $c_i$ and $c_j$. If the data distribution changes and the original grouping result is no longer appropriate, the grouping strategy can be reused to calculate the cosine similarity of data features. For example, if a certain category of data becomes more important in a new situation or a new data category appears, related devices may be regrouped into a more appropriate group.

$$\mathrm{S}_{ij} = \frac{h_i \cdot h_j}{\|h_i\| \, \| h_j\|} = \frac{\sum_{k=1}^{n} h_{ik} h_{jk}}{\sqrt{\sum_{k=1}^{n} h_{ik}^2}\sqrt{\sum_{k=1}^{n} h_{jk}^2}} \tag{1}$$

**Algorithm 1** Homogeneous Device Grouping.

```
Input: $C, G, H, P$;
Output: Grouping result $G$;
 1: Initialize $G \leftarrow \{\}, H \leftarrow \{\}$;
 2: for $i = 1$ to $n$ do
 3:     for $j = 1$ to $P$ do
 4:         Device $c_i$ pre-trains using its own data $d_i$;
 5:     end for
 6:     Extract feature map $v_i$ from ReLU layer;
 7:     Add to $H$: $h_i$ = PHOTON-Beetle-Hash($v_i$);
 8: end for
 9: Compute the cosine similarity between different devices' data distributions and save the result
    in the similarity matrix $M$, where $M_{ij} = \cos(\theta_{ij})$ and $\theta_{ij}$ is the angle between $H_i$ and $H_j$:
        $M_{ij} = \frac{H_i \cdot H_j}{\|H_i\|\|H_j\|}$
10: Edge server gets grouping result $G$ using improved ISODATA grouping on $M$: $G \leftarrow$
    ISODATA($M$);
11: return Grouping result $G$.
```

Improvements are made on the basis of iterative self-organizing data analysis technique (ISODATA) algorithm to enable grouping based on the cosine similarity matrix and classify devices with similar data distribution into the same group, as shown in Algorithm 1. The improved strategy is as follows:

a) Initialization: $K$ devices are randomly selected as the initial grouping center, and the grouping is divided into $\{c_1, c_2, \ldots, c_k\}$, where $c_i$ is the edge device.

b) Device allocation: For each edge device $c_i$, utilize its cosine similarity with each group center $c_j$ and assign it to the group $G_j$ with the highest similarity.

c) Update the grouping center: For group $G_j$, the steps of center update are as shown in formula (2), where $c'_j$ is the new center device of group $G_j$.

$$c_j^{'} = \underset{c_i \in G_j}{argmax} \left( \frac{1}{\left|G_j\right|} \sum_{c_k \in G_j} \mathrm{S}_{ik} \right) \tag{2}$$

d) Calculate similarity and standard deviation: By calculating the intra-group similarity $\mu_j$ and the standard deviation $\sigma_j$, the similarity between devices within the group and the degree of variation in the similarity within the group can be evaluated. If the $\mu_j$ of a group is very low and $\sigma_j$ is very high, it means that there is a large difference within the group, and it may be necessary to split it to form more homogeneous subgroups. The intra-group similarity $\mu_j$ of group $G_j$ can be obtained by formula (3), and the calculation formula of standard deviation $\sigma_j$ is as shown in (4).

$$\mu_j = \frac{1}{\left|G_j\right|^2 - \left|G_j\right|} \sum_{c_i \in G_j} \sum_{c_k \in G_j, k \neq i} \mathrm{S}_{ik} \tag{3}$$

$$\sigma_j = \sqrt{\frac{1}{\left|G_j\right| - 1} \sum_{c_i \in G_j} \left( \frac{\sum_{c_k \in G_j, k \neq i} \mathrm{S}_{ik}}{\left|G_j\right| - 1} - \mu_j \right)^2} \tag{4}$$

In formula (3), the double summation $\sum_{c_i \in G_j} \sum_{c_k \in G_j, k \neq i}$ represents the calculation of the total sum of similarities between all possible distinct device pairs $(c_i, c_k)$ within the group $G_j$. The denominator $|G_j|^2 - |G_j|$ indicates the total number of all possible unique device pairs within the group $G_j$. $|G_j|$ is subtracted from $|G_j|^2$ (all possible pairings among devices, including self-pairings) as the self-pairings of devices do not need to be calculated.

If the center similarity $\mu_{jl}$ of two groupings is high and the grouping size is small, then the grouping centers may be too close and can be merged to form a larger, more representative grouping. The calculation formula of similarity $\mu_{jl}$ between groups is shown in (5).

$$\mu_{jl} = \frac{1}{\left|G_j\right| \cdot |G_l|} \sum_{c_i \in G_j} \sum_{c_k \in G_l} \mathrm{S}_{ik} \tag{5}$$

e) Split and merging:

Split operations are triggered when there is substantial variation in the data distribution within a group. Specifically, a group $G_j$ experiences a split process when the internal average similarity $\mu_j$ falls below the threshold $T_{\mu,\text{split}}$ and the standard deviation $\sigma_j$ exceeds the threshold $T_{\sigma,\text{split}}$. These conditions indicate that the data distribution within the group is not sufficiently uniform, necessitating further subdivision to achieve more homogeneous subgroups.

Merging operations are considered when the centers of two groups are very close to each other. A merge between groups $G_j$ and $G_l$ is executed when the inter-group center similarity $\mu_j l$ exceeds the threshold $T_{\mu,\text{merge}}$ and the sizes of both groups are below the threshold $T_{size,\text{merge}}$. The objective of merging operations is to reduce excessive split among groups, thereby enhancing the overall efficiency of the system.

These processes are integral to maintaining optimal group configurations in systems that utilize FL, ensuring that each group is as homogeneous as possible with regard to data distribution among its devices.

f) Iteration and output: Repeat steps b) to e) until any of the following termination conditions are met: the change of each group member is less than the threshold, or each group reaches the preset target similarity, or the maximum number of iterations is reached. Finally, the output grouping result $G = \{G_1, G_2, \ldots, G_K\}$, all devices in each group are isomorphic, that is, they have similar data distribution.

**Algorithm 2** CMAB-based Client Scheduling Strategy.

```
Input: State report $S_i = \{U_i, N_i\}$ for each device $c_i$, $U_i$ represents the computational resource situation, while $N_i$ signifies the network resource situation, Reward function $f(\cdot)$, Number of top devices to select $m$, Number of rounds $T_{ics}$;
Output: Selected device set $V$ for all rounds;
1: Initialize $V \leftarrow \{\}$;
2: for $t = 1$ to $T_{ics}$ do
3:     for each device $c_i$ in $C$ do
4:         Compute expected reward $Q_{i,t} = f(S_{i,t}) + \epsilon_{i,t}$;
5:     end for
6:     Select $m$ devices with the highest $Q_{i,t}$ and denote them as $V_t = \{c_{i1}, c_{i2}, \ldots, c_{im}\}$;
7:     Add $V_t$ to the set $V$;
8: end for
9: return $V$.
```

**Algorithm 3** Intra-Group Federated Learning.

```
Input: $C, K, E$;
Output: Each intra-group model $W_k$ trained by GSFL;
1: Call Algorithm 1 to get group result $G$;
2: for $k = 1$ to $K$ in parallel do
3:     for $l = 1$ to $E$ do
4:         Call Algorithm 2 to get scheduled client set $I_k$;
5:         for Each client $c_i \in I_k$ in parallel do
6:             Client $c_i$ trains the received model locally for round $l$:
                   $$W_i^{l+1} = W_i^l - \frac{\alpha}{|d_i|} \sum_{j=1}^{|d_i|} \nabla f\left(W_i^l, x_{i,j}, y_{i,j}\right)$$
7:             Client $c_i$ uploads local model $W_i^{l+1}$ to edge server;
8:         end for
9:         Edge server aggregates uploaded models from selected clients within group into new intra-group model:
                   $$W_k^{l+1} = \frac{\sum_{c_i \in G_k} |d_i| W_i^{l+1}}{\sum_{c_i \in G_k} |d_i|}$$
10:    end for
11: end for
12: return Each intra-group model $W_k$.
```

*3.1.3 Intra-group client scheduling model.* CMAB is based on the traditional Multi-armed Bandit (MAB) and adds the element of "context". That is to say, at each decision-making moment, in addition to choosing the arm of the slot machine, there will be some additional relevant information (i.e., context) available, such as the characteristics of the user, the state of the environment, etc. This contextual information can assist the player in making wiser decisions to enhance the possibility of obtaining higher returns.

Through the intra-group client scheduling module, the CMAB strategy is adopted to select clients for training according to the computing and network resource conditions of the device. This avoids the increased delay caused by over-involvement of resource-constrained devices in training and at the same time ensures the efficiency and reliability of training.

When the device situation changes, for example, when the computational capability of the device changes due to hardware aging or software updates, or when network resources fluctuate due to changes in the network environment, the device will report its new status to the server. The server recalculates the expected reward according to the new status and adjusts the selection of clients. For devices with reduced capabilities, their frequency of participating in training may be reduced. For devices with improved capabilities, their opportunities to participate in training may be increased, thus adaptively adjusting the training process.

In the scenarios configured in this article, each device group is treated as an arm in the CMAB problem, and a subset of clients is selected for FL training at each action round. The client scheduling strategy adopted by the intra-group client scheduling model is shown in Algorithm 2. At the start of each round $t$, devices within each group transmit status reports $S_{i,t} = \{U_{i,t}, N_{i,t}\}$ to the server, detailing its own computing resources $U_{i,t}$ and network resources $N_{i,t}$. In response, the server implements a CMAB-based client scheduling policy based on these reports.

Specifically, the server calculates the expected reward $Q_{i,t}$ of each device in the group, and selects the top m devices as the client set $V_t = \{c_1, c_2, \ldots, c_m\}$. These selected devices participate in FL training as clients, receiving the corresponding intra-group models from the server for training on local data. The trained gradients are returned to the server for aggregation and intra-group model updates.

The expected reward $Q_{i,t}$ is calculated from the device resource status and a small random number $\epsilon_{i,t}$, which can be learned from historical data, as shown in (6). The reward function is given by formula (7), $\omega_1$ and $\omega_2$ are the importance weights of the two resources for the reward. The computing resource utility function $g\left(U_{i,t}\right)$ is designed as a Logistic function with a limit value to simulate the diminishing marginal effect of increasing computing resources on performance improvement, as shown in formula (8), where $\alpha$ represents the saturation upper limit, i.e., the maximum utility value. $\kappa$ is the growth rate, which determines how quickly the function grows. $U_{\text{mid}}$ is the turning point at which the resource's utility grows from fast to slow growth, that is, the midpoint of the function, set to the median of the computing resources expected to be used. The network resource utility function $h\left(N_{i,t}\right)$ is designed as a Logistic function with a soft saturation upper limit, as shown in formula (9), where $\beta$ is the maximum contribution of network resources to utility, $\lambda$ is the rate at which network resource utility increases. The inflection point method is used to analyze performance indicators and network resource data, and the point where the second-order derivative is zero is found as the saturation point $N_{\text{sat}}$ of network resources. When network resources reach this value, utility growth will become very slow. The CMAB strategy dynamically adjusts the selection of clients according to $g\left(U_{i,t}\right)$ and $h\left(N_{i,t}\right)$. When a certain device fails, the server can select other appropriate devices for training according to the status reports of other devices, avoiding the stagnation of the entire training process due to individual device failures and ensuring the stability of the system.

$$Q_{i,t} = f\left(S_{i,t}\right) + \epsilon_{i,t} \tag{6}$$

$$f\left(S_{i,t}\right) = \omega_1 \cdot g\left(U_{i,t}\right) + \omega_2 \cdot h\left(N_{i,t}\right) \tag{7}$$

$$g\left(U_{i,t}\right) = \frac{\alpha}{1 + e^{-\kappa \cdot \left(U_{i,t} - U_{\text{mid}}\right)}} \tag{8}$$

$$h\left(N_{i,t}\right)=\frac{\beta\cdot\left(1-e^{-\lambda\cdot N_{i,t}}\right)}{1+e^{-\lambda\cdot\left(N_{i,t}-N_{\text{sat}}\right)}} \tag{9}$$

The server collects rewards, calculates the time difference error, and updates the parameter vector $\delta_t$ using formula (10), where $r_{t+1}$ is the actual reward obtained at time $t+1$, and $\gamma$ is the discount that determines the importance of future rewards. The factors, $Q_{i,t}$ and $Q_{i,t+1}$ are the expected rewards at time $t$ and $t+1$ respectively. During the entire process, the cumulative regret $R_T$ of each round is calculated to measure the algorithm performance. The accumulated regret $R_T$, which is the sum of regrets in all rounds, reflects the suboptimal selection of device sets. The calculation formula is (11), where $R_T$ represents the accumulated regret in $T$ rounds, indicating the total performance loss incurred due to not selecting the optimal device set. $Q_{*,t}$ represents the maximum reward that may be obtained at time $t$, estimated using the $\varepsilon$-greedy strategy. Specifically, $Q_{*,t}$ is the maximum reward observed up to the current time, while $Q_{V_t,t}$ represents the reward obtained by selecting $V_t$ at time $t$. The goal of intra-group device scheduling is to minimize accumulated regret $R_T$, thereby improving the performance of the FL system and making optimal use of the computing and network resources of each device. The specific intra-group FL process is presented in Algorithm 3.

$$\delta_t=r_{t+1}+\gamma Q_{i,t+1}-Q_{i,t} \tag{10}$$

$$R_T=\sum_{t=1}^{T}\left(Q_{*,t}-Q_{V_t,t}\right) \tag{11}$$

## 3.2 Grouping-Split federated learning

Grouping FL based on the SL mechanism mainly addresses the problems of weak privacy protection and high device load in FL, and focuses on solving the impact of device heterogeneity. In traditional FL, participants need to share original model or gradient information when performing model aggregation, which may lead to the risk of privacy leakage. Grouping FL uses a SL mechanism to divide the holistic model according to a hierarchical structure, so that participants only need to transmit the model parameters or features within their respective groups without sharing the holistic model or gradient information. Both the client and the server can only access a part of the entire model, thereby reducing the risk of client or server privacy leakage due to reconstruction attacks and effectively protecting privacy.

Device heterogeneity is also a focus of grouping FL. The server divides each device into different groups based on the similarity of its data distribution. After completing the device grouping, perform channel pruning on the device-side model and server-side model to remove unimportant channels and reduce the amount of parameters and calculations. The models within each group maintain a consistent hierarchical structure and number of channels. In this way, the impact of model heterogeneity introduced by the SL mechanism is significantly reduced. This enables more effective FL model aggregation, leading to overall model improvement and enhanced performance of the holistic model.

Fig. 2 shows the overall framework of the privacy-preserving grouping FL method based on SL, which includes edge devices at the terminal level, federated servers at the edge level, and cloud servers at the cloud level. The figure shows the main workflow of the proposed method. Through device grouping, model decomposition, and channel pruning, grouping FL based on SL mechanism can improve privacy security, improve the effect of model aggregation, and improve the stability of aggregation results and the performance of the final model. At the same time, since the SL mechanism only needs to transmit the parameters or features of the model within the

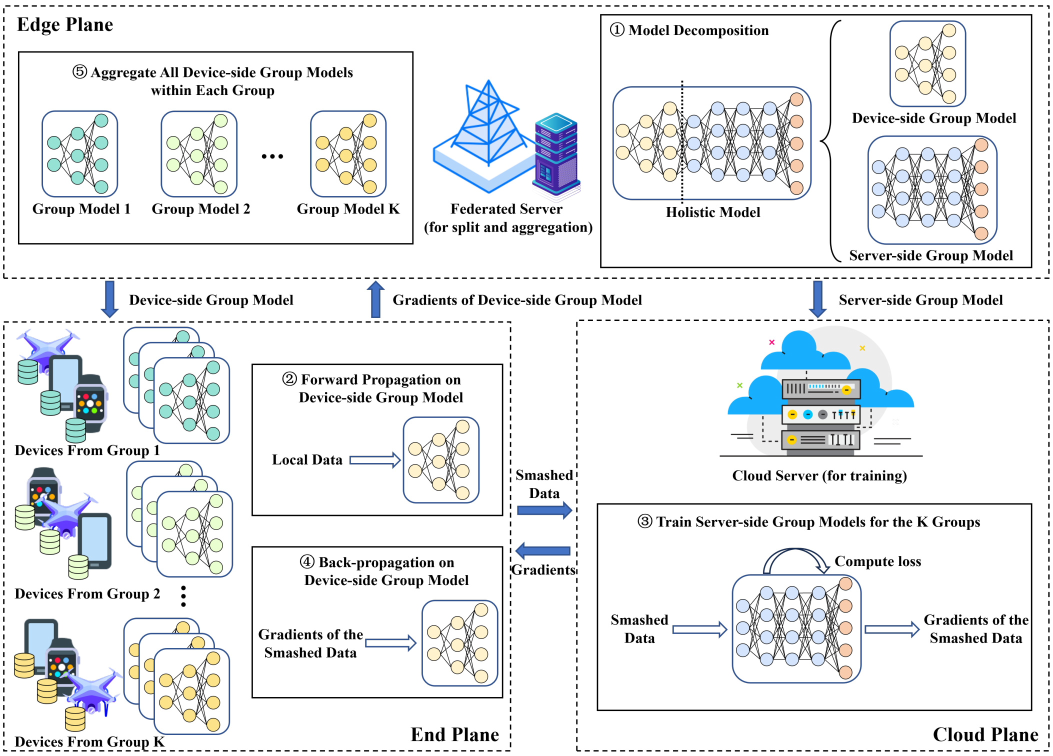

Edge Plane
⑤ Aggregate All Device-side Group Models within Each Group
Group Model 1
Group Model 2
Group Model K
Federated Server (for split and aggregation)
① Model Decomposition
Holistic Model
Device-side Group Model
Server-side Group Model
Device-side Group Model
Gradients of Device-side Group Model
Server-side Group Model
Devices From Group 1
Devices From Group 2
Devices From Group K
② Forward Propagation on Device-side Group Model
Local Data
④ Back-propagation on Device-side Group Model
Gradients of the Smashed Data
End Plane
Smashed Data
Gradients
Cloud Server (for training)
③ Train Server-side Group Models for the K Groups
Compute loss
Smashed Data
Gradients of the Smashed Data
Cloud Plane

Fig. 2. The overall framework diagram of GSFL.

group, rather than the parameters or features of the holistic model, communication overhead is reduced, and the holistic FL model is decomposed into the device part and the server part, which reduces resources. The computational load of constrained devices improves the efficiency of FL. This grouping FL method based on the SL mechanism is suitable for heterogeneous and resource-constrained scenarios. While maintaining the ability to process Non-IID data, it introduces a more powerful privacy protection method, optimizes the FL architecture, and enhances the privacy is protected and the load of the device is reduced, which has important practical application value.

*3.2.1 Model decomposition.* In the privacy-preserving grouping FL process based on the SL mechanism, each group has a unique group model. First, the federated server at the edge layer applies the model decomposition strategy to analyze the hierarchical structure of the CNN model. It calculates the computational complexity $O_l$ (measured in floating point operations, FLOPs) of each layer $l$ (the total number of layers is $L$), and the size of the output data $\mathcal{N}_l$ (the size of the activation value, i.e., the size of the feature map from one layer to the next). In the pre-training process necessary for the grouping strategy, record the accuracy $a_i$ and loss value $l_i$ of each device $i$ in each round of pre-training, and calculate the standard deviation of the accuracy $\sigma_{a_i}$ and the standard deviation of the loss value $\sigma_{l_i}$. Define the device performance score $p_i$ as (12). The higher the device performance score, the smaller the performance fluctuation of the device during the pre-training process and the more stable the performance. Use the exponential function $e^{-\sigma_{a_i}}$ to process the standard deviation of accuracy so that the smaller the accuracy fluctuation, the higher the score. Use the logarithmic function $\ln(1+\sigma_{l_i})$ to process the standard deviation of loss value and place it in the denominator position so that the greater the loss value fluctuation, the lower the score.

$$p_i = \frac{e^{-\sigma_{a_i}}}{1+\ln(1+\sigma_{l_i})} \tag{12}$$

The average computational capability of edge devices within group $k$ is denoted as $\overline{\Psi}_k$, as shown in formula (13). Using formula (14), among all layers whose computational complexity does not exceed the device capability, select the layer with the smallest output data size as the split point $sp$, and divide each group model into a device-side group model and a server-side group accordingly. Here, $\tau$ is a hyperparameter between 0 and 1, used to control the position of the split point. When $\tau$ is small, the split point will be closer to the shallow part of the CNN. Conversely, when $\tau$ is large, the split point will be closer to the deep part of the CNN. The parameter $\tau$ can be fine-tuned to precisely control the placement of the split point inside the CNN architecture, thus preventing it from being too close to the network's initial or shallow layers.

$$\overline{\Psi}_k = \frac{\sum_{i=1}^{M} p_i \times \Psi_{device_i}}{\sum_{i=1}^{M} p_i} \tag{13}$$

$$sp = \arg\min_{l}\{\mathcal{N}_l \mid O_l \leq \overline{\Psi}_k \text{ and } l \in \{1,2,\ldots,L\}\ ,\ l \geq \tau L\} \tag{14}$$

Model decomposition divides the group model into two parts: the device-side group model and the server-side group model. This ensures that the edge device will not become slow due to excessive computational load when executing the device-side model. Additionally, it aims to reduce the amount of data transmitted over the network, thereby reducing communication latency. Participants within each group are only responsible for training and updating the local device-side group model, which is usually one or more layers of the holistic group model. Participants only use local data to train their intra-group parts, without sharing raw data with other participants. The cloud server is responsible for maintaining and updating the server-side group model for each group, which typically contains the remaining layers and parameters of the group model. During the training process, the cloud server receives model updates within the group from each group and integrates these updates into the server part.

*3.2.2 Adaptive channel-level pruning.* In a heterogeneous and resource-constrained edge computing scenario, smart devices typically exhibit variations in their characteristics. After completing the device grouping, channel pruning needs to be performed on the device-side model and the server-side model to remove unimportant channels. Reduce the amount of parameters and calculations to reduce the load pressure on resource-constrained edge devices. This makes the model more efficient in running on resource-constrained devices, reduces processing time, and ensures low latency and high reliability. The adaptive channel pruning strategy involved is only performed in the first communication round of FL, and will not change the structure of the device-side group model after channel pruning in the subsequent training process.

Each client trains and evaluates the model on a local dataset, and the federated server collects the model parameters of all clients and aggregates them. For each grouping, the $l_1$-norm of each channel in each convolutional layer of the global device-side group model is calculated as the channel importance score $I_c^{(i)}$, which is used for the evaluation of channel importance, as shown in formula (15), where $w_c^{(i)}$ represents the weight vector of the $c$-th channel of the $i$-th layer. Based on the calculated importance score $I_c^{(i)}$, all channels are sorted in order to identify the channel with the lowest importance. According to the set pruning proportion $p_k$, determine a threshold $T_{imp_k}$ such that among all channels, the importance score of a proportion $p_k$ of channels is lower than $T_{imp_k}$. In this context, $p_k$ denotes the predefined pruning ratio for the model associated with group $k$. The cloud server creates a mask matrix $\mathbf{M}^{(i)}$ for the convolution layer $i$ for channel pruning, so that the mask matrix corresponds to the element $M_c^{(i)}$ of the $c$-th channel as shown in (16). Finally, the Hadamard product is calculated using the weight matrix and the mask matrix, that is, element-wise multiplication, as shown in formula (17), each weight value will be multiplied by the corresponding element (0 or 1) in the mask matrix. If the value in the mask is 0, the corresponding weight will be set to 0 (i.e. pruned). If the value in the mask is 1, the corresponding weight remains unchanged. This approach achieves selective pruning of channels in convolutional layer $i$, removing less important channels. For the global server-side model, perform the same channel pruning operation.

$$I_c^{(i)} = \left\|\mathbf{w}_c^{(i)}\right\|_1 \tag{15}$$

$$M_c^{(i)} = \begin{cases} 1, & if\ I_c^{(i)} > T_{imp} \\ 0, & otherwise \end{cases} \tag{16}$$

$$\mathbf{w}^{(i)} \leftarrow \mathbf{w}^{(i)} \odot \mathbf{M}^{(i)} \tag{17}$$

*3.2.3 Grouping federated learning process based on split learning mechanism.* All devices participating in FL undergo pre-training and extract data distribution characteristics using lightweight hash encryption function. These characteristics are then sent to the federated server. The federated server collects the data distribution characteristics of all devices and completes device grouping based on this. When training starts, the federated server initializes the global device-side group model $W_t^C$ and server-side group model $W_t^S$ according to the model decomposition strategy, and then selects a subset of clients from the groups of individual clients. At the beginning of each round, each client selected to participate in training downloads $W_t^C$ from the federated server and then uses its local data in parallel to perform forward propagation on the local device-side group model until the split layer is reached. Afterwards, the shredded data is sent to the cloud server, which also downloads $W_t^S$ from the federated server and completes forward propagation and backward propagation on the corresponding server-side group model. Subsequently, the cloud server feeds back the gradient of the smashed data to the clients in each group to complete the backpropagation process of the device-side group model. At the end of each round, the client updates $W_t^C$, which is then sent to the federated server for aggregation. Typically, the federated server can be a trusted third party or authority that only assists in coordinating the entire training process and implementing the aggregation of device-side group models. Therefore, unlike cloud servers, federated servers do not necessitate high-performance computing resources. The specific implementation of GSFL is illustrated in Algorithm 4.

## 4 EXPERIMENTS

### 4.1 Experimental setting

*4.1.1 Dataset and model.* The experiments employed datasets from Fashion-MNIST [45] and MNIST [3], each comprising 10 distinct categories. To simulate data heterogeneity under a Non-IID scenario, the data for each client was sourced from a Dirichlet distribution, Dir($\alpha$). This approach models varying degrees of data heterogeneity and imbalanced label distribution. A larger $\alpha$ value signifies a balanced distribution among categories, approaching an IID setting, whereas a smaller $\alpha$ yields a noticeably heterogeneous data distribution. The first step is to determine the number of categories to simulate, which defines the dimensions of the Dirichlet distribution. Next, an $\alpha$ vector is established, where uniform values indicate an even distribution of categories, and smaller

**Algorithm 4** Implement of Grouping-Split Federated Learning (GSFL).

```
Input: $\mathcal{T}$: Number of FL communication rounds. $W_{k,i}^{(t)}$: Device-side model for client $i$ in group $k$
    at round $t$. $W_k'^{(t)}$: Server-side model for group $k$ at round $t$.
Output: Device-side model $W_k$ and server-side model $W_k'$ for each group $k$.
 1: Call Algorithm 1 to get group result $G$.
 2: for each group $k \in G$ do
 3:     Parameter server initializes device-side model $W_{k,i}^{(0)}$ and server-side model $W_k'^{(0)}$ according
        to the model decomposition strategy.
 4: end for
 5: for $t = 1$ to $\mathcal{T}$ do
 6:     for each group $k \in G$ do
 7:         The parameter server calls Algorithm 2 to select a subset $V$ of clients within the group.
 8:         for each client $i \in V$ do
 9:             Client $i$ performs forward propagation on $W_{k,i}^{(t)}$ using local data $d_i$ up to the split layer
                $sp$.
10:             Client $i$ sends the activation maps at the split layer to the cloud server.
11:         end for
12:         Cloud server downloads the server-side model $W_k'^{(t)}$ from the parameter server, and
            completes forward propagation and backpropagation on $W_k'^{(t)}$.
13:         Cloud server sends the gradients back to the clients in group $k$.
14:         for each client $i$ scheduled within group $k$ do
15:             Client $i$ completes backpropagation on $W_{k,i}^{(t)}$ using received gradients.
16:             Client $i$ sends the updated parameters to the parameter server.
17:         end for
18:         Parameter server aggregates the updates from the participating clients within each group
            to update $W_{k,i}^{(t)}$ and $W_k'^{(t)}$ using (18) and (19).
```

$$W_{k,i}^{(t+1)} = W_{k,i}^{(t)} - \frac{\alpha}{|d_i|} \sum_{j=1}^{|d_i|} \nabla f(W_{k,i}^{(t)}, x_{ij}, y_{ij}) \tag{18}$$

$$W_k'^{(t+1)} = \frac{\sum_{c_l \in G_k} |d_i| W_{k,i}^{(t+1)}}{\sum_{c_l \in G_k} |d_i|} \tag{19}$$

```
19:     end for
20: end for
21: return Device-side model $W_k$ and server-side model $W_k'$ for each group $k$.
```

values suggest a skewed distribution. Each client extracts a sample from the predefined Dirichlet distribution, with each vector element representing the proportion of a specific category within that client's data. These proportions are used to generate the corresponding category labels for each client. By adjusting the $\alpha$ value, a range of category distribution scenarios can be simulated, from balanced to imbalanced.

Specific $\alpha$ values of 1.0, 0.5, 0.1, and 0.05 have been selected for the experiments. These selected $\alpha$ values allow for a comprehensive investigation and demonstration of the effects of different degrees of data heterogeneity on the experimental results, providing a deeper understanding of the performance and behavior of the proposed method in various data distribution scenarios.

- When $\alpha$ is set to 1.0, the data distribution among categories is relatively balanced, closely resembling an IID setting. This indicates that each category has a relatively equal likelihood of occurrence. For instance, in the Fashion-MNIST dataset, the frequency of different clothing items is approximately the same.
- When $\alpha$ is 0.5, a moderate degree of heterogeneity emerges. Categories begin to exhibit some imbalance, yet not to a significant extent. Consider the MNIST dataset; some digit classes might occur more frequently than others, but the disparity is not overly pronounced.
- With $\alpha$ values of 0.1 and 0.05, the data distribution becomes highly heterogeneous. In such circumstances, certain categories might predominate the data for a specific client, while others are scarce. For example, in a client's data from the Fashion-MNIST dataset, one type of clothing might constitute a large proportion, and others might be present only in small quantities.

The model architecture employed is MobileNetV2 [34], a lightweight CNN designed specifically for mobile devices and well-suited for edge computing. Given the resource constraints of FL clients in resource-constrained scenarios, the smallest variant, MobileNetV2-0.25, was specifically utilized. This model variant is particularly apt for edge computing in scenarios where resources are limited.

*4.1.2 Comparison methods.* In this paper, the comparative methods selected include the FL baseline algorithm FedAvg, the grouping FL method HACCS which protects user data distribution characteristics using DP techniques, the newly defined SplitFed approach to split FL, and Cluster-HSFL, which combines grouping with the SL mechanism.

FedAvg [31]: This method involves multiple clients collaboratively training a shared global model without sharing the original data. In the scenario set up for this paper, the global model is tested on all clients to assess performance.

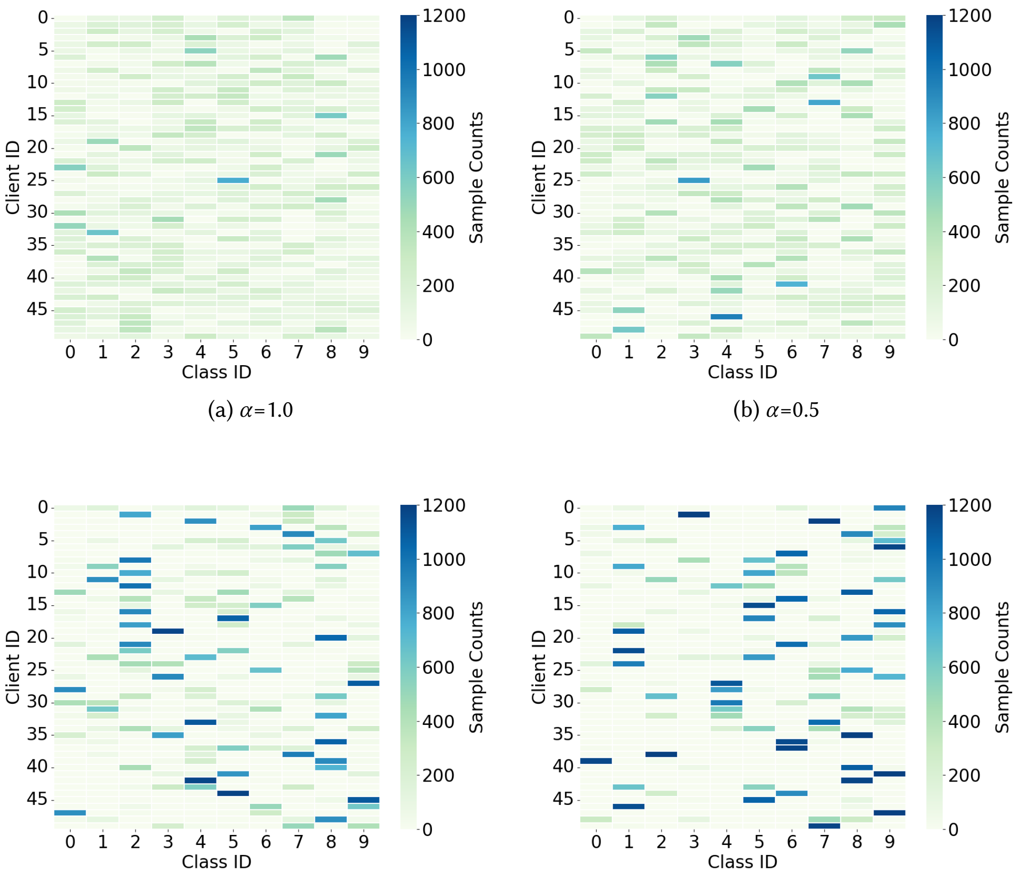
HACCS [43]: HACCS is a grouping FL method that accounts for data heterogeneity. Its core concept groups similar clients into different groups to facilitate local training within each group. This reduces the issue of weight divergence between groups, lowers communication overhead, and results in a high-performance personalized FL model.

SplitFed [41]: SplitFed addresses the inherent drawbacks of slower execution in SL compared to FL, and weaker privacy protection in FL as opposed to SL. By partitioning the model into multiple parts and training them separately on the client and server, SplitFed significantly reduces the computational demands on clients. This makes it suitable for resource-constrained devices and enhances data privacy by integrating DP.

Cluster-HSFL [52]: The algorithm combines grouping, SL, and FL to augment the parallelism inherent in model training and to attenuate the computational burden imposed on clients. It mitigates the workload on individual clients by partitioning the model, thereby diminishing the latency associated with training. Concurrently, it enhances privacy protection.

## 4.2 Experiment results

In the context of FL, the use of grouping can improve the accuracy of models. Fig. 3 demonstrates client data distribution under different degrees of data distribution heterogeneity. The shades of color represent the amounts of corresponding category data that different clients possess. The darker the color, the greater the amount of data the client corresponding to the square holds for the corresponding category. When the $\alpha$ value is large, it implies a balanced distribution among categories, approaching the IID setting. At this point, the color distribution in the figure is relatively more balanced, and the color difference of each square is relatively small. When the $\alpha$ value is small, an obvious heterogeneous data distribution will occur. In this case, the color of some squares in the figure will be significantly darker, while that of most squares will be very pale, which intuitively reflects the highly uneven distribution of data among different clients. Fig. 4 details the test accuracy over 300 communication rounds under varying degrees of data distribution heterogeneity. The introduction of a SL mechanism can also enhance privacy security and reduce the training latency of the FL model. Table 2 shows the comparison results of training latency on the MNIST dataset for the proposed GSFL algorithm and various other methods when the heterogeneity measure $\alpha$ is set to 0.5.

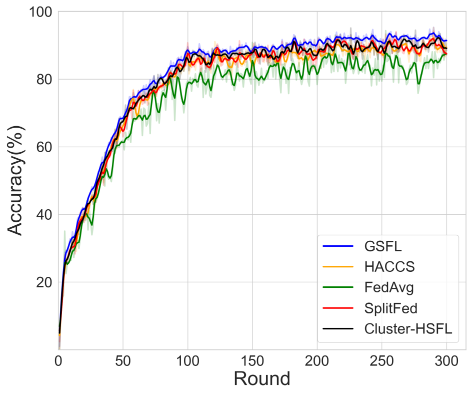


(a) $\alpha$=1.0 (b) $\alpha$=0.5

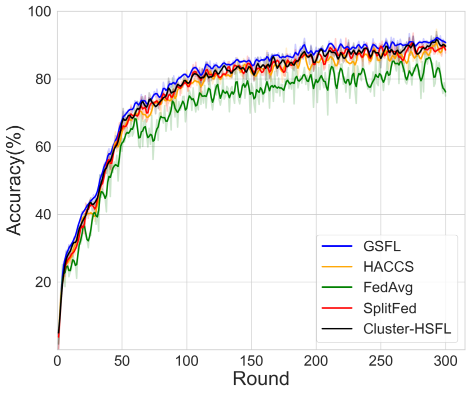
(c) $\alpha$=0.1 (d) $\alpha$=0.05

Fig. 3. Illustration of client data distribution under different degrees of data distribution heterogeneity.

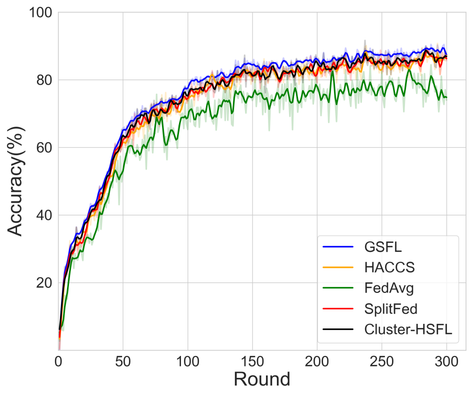
According to the results in Fig. 4, the GSFL algorithm demonstrates superior performance over FedAvg, HACCS, SplitFed, and Cluster-HSFL across various settings of the Dirichlet distribution parameter ($\alpha$). This advantage is particularly pronounced under Non-IID conditions, where GSFL consistently achieves the highest accuracy. When $\alpha$ is set to 1.0, the data distribution is relatively uniform, approximating an Independent and Identically Distributed (IID) scenario. In this scenario, GSFL exhibits a significant performance edge. Specifically, GSFL shows an improvement of 2.39% over FedAvg, 1.56% over HACCS, and 0.76% over Cluster-HSFL. These results underscore GSFL's



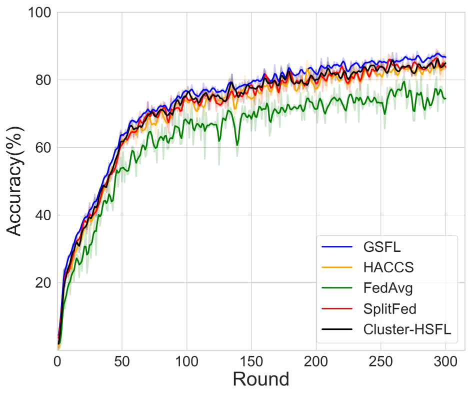
(a) $\alpha$=1.0



(b) $\alpha$=0.5



(c) $\alpha$=0.1



(d) $\alpha$=0.05

Fig. 4. Test accuracy (%) under various degrees of data distribution heterogeneity within 300 communication rounds.

ability to enhance FL efficiency and model performance through its optimized client scheduling strategy, even in scenarios with a more uniform data distribution. As $\alpha$ decreases to 0.05, the data distribution becomes highly skewed, simulating an extreme Non-IID scenario. Under these conditions, the performance advantage of GSFL becomes even more evident. GSFL exhibits an improvement of 10.64% over FedAvg and 4.53% over HACCS, highlighting its robust capability in handling extreme Non-IID data distributions. Compared to SplitFed and Cluster-HSFL, the improvements are 2.29% and 1.16%, respectively, further demonstrating GSFL's effectiveness in optimizing data heterogeneity.

The success of GSFL in terms of accuracy is due to its strategic client scheduling method, effectively handling unique data distributions in Non-IID conditions. In contrast, FedAvg encounters difficulties capturing these distributions, leading to a noticeable decline in performance. Moreover,

Table 2. Training delay performance comparison among different methods on the MNIST dataset.

| Accuracy | Delay(s) | | | | |
|---|---|---|---|---|---|
| | GSFL | Cluster-HSFL [52] | HACCS [43] | SplitFed [41] | FedAvg [31] |
| 25% | 8.31 | 9.37 | 11.79 | 12.31 | 36.65 |
| 50% | 25.89 | 27.75 | 28.53 | 28.36 | 89.93 |
| 75% | 39.25 | 41.97 | 44.15 | 44.75 | 193.53 |
| 80% | 51.46 | 55.14 | 57.91 | 58.32 | 236.72 |
| 85% | 73.84 | 79.38 | 84.67 | 85.59 | 349.27 |
| 90% | 97.66 | 102.91 | 112.35 | 115.71 | 637.51 |

GSFL's strategy surpasses HACCS's random device selection method, highlighting GSFL's effectiveness in managing Non-IID conditions and optimizing the outcomes of FL. Therefore, despite HACCS's focus on client personalization, GSFL demonstrates superior overall performance. Compared to SplitFed and Cluster-HSFL, which also introduce SL mechanisms, GSFL adds a lightweight homogenous device grouping module, which contributes to an increase in accuracy to a certain extent. The SL mechanism incorporated in GSFL plays a crucial role in enhancing the participation capabilities of resource-constrained devices in FL training. It reduces the computational burden and data transmission requirements for these devices, enabling them to actively contribute without being hindered by their limitations.

According to the results in Table 2, GSFL consistently outperforms other methods in terms of latency performance. Notably, when achieving an accuracy of 90%, GSFL reduces training time by 6.53× compared to FedAvg. The latency reductions compared to SplitFed and HACCS are also significant, usually ranging between 1.10× and 1.18×. Relative to Cluster-HSFL, the reduction is smaller but still between 1.05× and 1.13×. It should be noted that the random scheduling scheme designed in the FedAvg method brings about a serious straggler problem during the FL training process. This is manifested in the significantly higher latency required to reach the specified accuracy compared to other methods, including GSFL. The straggler problem in FedAvg leads to inefficiencies and prolonged training times, which negatively affects the overall performance and scalability of the FL system. By contrast, GSFL and the other compared methods have addressed this problem to varying degrees, contributing to their superior performance in terms of latency and training efficiency.

These results indicate that both the grouping FL method HACCS and the split FL method SplitFed can improve the training efficiency of FL. Combining grouping with SL further enhances training efficiency while also bolstering privacy protection to some degree. Compared to the similarly conceptualized Cluster-HSFL method, the method proposed in this paper is still somewhat superior. This is because it combines the designed lightweight client scheduling method with the SL mechanism, avoiding the extensive computational resource consumption that comes with encryption technologies such as homomorphic encryption and DP. As a result, GSFL can improve training efficiency and reduce device load in scenarios constrained by computational and storage resources, which is reflected in the lowest FL training latency of GSFL compared to other methods. Furthermore, the introduced SL mechanism provides enhanced model privacy over traditional grouping FL methods.

Overall, GSFL has consistently demonstrated outstanding performance across a wide range of scenarios. In both uniform and highly skewed data distributions, it outperforms competing methods, showcasing its adaptability and effectiveness. When it comes to latency, GSFL significantly reduces training time, addressing the straggler problem and improving overall training efficiency. The addition of the lightweight homogenous device grouping module and the strategic client scheduling method, along with the SL mechanism, allows GSFL to handle data heterogeneity, enhance model privacy, and optimize the outcomes of federated learning. Whether in scenarios with ample resources or those constrained by computational and storage limitations, GSFL proves to be a superior solution, offering enhanced performance and potential for widespread application in diverse federated learning contexts.

## 5 CONCLUSION

This study introduces an innovative FL approach named GSFL, designed to address the challenges of data privacy protection and heterogeneity in resource-constrained edge computing scenarios. GSFL efficiently enhances the training efficiency by integrating a grouping strategy with a SL mechanism, reduces the load on resource-constrained devices, and strengthens data privacy, especially under Non-IID data conditions. Experimental results indicate that GSFL outperforms conventional FL algorithms such as FedAvg, the DP-incorporating grouping FL method HACCS, the split FL method SplitFed, and the grouping and SL integrated method Cluster-HSFL on the Fashion-MNIST and MNIST datasets. Specifically, GSFL achieves the highest accuracy under Non-IID data conditions and exhibits superior training efficiency compared to other algorithms, demonstrating its effectiveness in handling Non-IID scenarios and optimizing FL outcomes. The strategic client scheduling method employed by GSFL effectively manages the unique data distributions inherent in Non-IID conditions, as opposed to mere random device selection methods. Moreover, GSFL's lightweight homogenous device grouping module further improves accuracy and training efficiency, while avoiding the use of computationally intensive encryption technologies such as homomorphic encryption and DP. Therefore, in scenarios constrained by computational and storage resources, GSFL helps to reduce the burden on the device.

In summary, the GSFL algorithm meets data privacy protection needs while effectively addressing the issues of data and device heterogeneity in edge computing contexts, offering an efficient FL solution suitable for large-scale, resource-constrained scenarios. Its innovation and practicality are anticipated to advance the widespread application of FL technology in the field of mobile multimedia communication, particularly in the deployment of edge autonomous systems. However, this study is not without its limitations. Although GSFL has shown promise in controlled experimental settings, its scalability to larger and more complex networks, encompassing thousands of devices, remains to be evaluated. As network sizes increase, the complexity of client scheduling and data processing might greatly increase. Moreover, the success of GSFL largely depends on the effective grouping of devices based on data characteristics, which may not always be feasible or accurate in real-world scenarios where data dynamics are constantly changing. Future work could focus on developing more robust and scalable client selection algorithms that can handle larger federation of devices without substantial trade-offs in performance or privacy. Additionally, to better manage the dynamic nature of edge devices and data distribution, adaptive grouping strategies could be developed, which would periodically reevaluate and readjust device groupings based on ongoing data characteristics.

## ACKNOWLEDGMENTS

This work has received funding from National Natural Science Foundation of China (No. 42275157).

## REFERENCES

[1] Christopher Briggs, Zhong Fan, and Peter Andras. 2020. Federated learning with hierarchical clustering of local updates to improve training on non-IID data. In *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE,

Glasgow, UK, 1–9. https://doi.org/10.1109/IJCNN48605.2020.9207469

[2] Luca Corinzia, Ami Beuret, and Joachim M Buhmann. 2019. Variational federated multi-task learning. *arXiv preprint arXiv:1906.06268* (2019). https://doi.org/10.48550/arXiv.1906.06268

[3] Li Deng. 2012. The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]. *IEEE Signal Processing Magazine* 29, 6 (2012), 141–142. https://doi.org/10.1109/MSP.2012.2211477

[4] Ruijun Deng, Xin Du, Zhihui Lu, Qiang Duan, Shih-Chia Huang, and Jie Wu. 2023. HSFL: Efficient and Privacy-Preserving Offloading for Split and Federated Learning in IoT Services. In *2023 IEEE International Conference on Web Services (ICWS)*. IEEE, Chicago, Illinois, USA, 658–668. https://doi.org/10.1109/ICWS60048.2023.00084

[5] Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. 2020. Adaptive personalized federated learning. *arXiv preprint arXiv:2003.13461* (2020). https://doi.org/10.48550/arXiv.2003.13461

[6] Enmao Diao, Jie Ding, and Vahid Tarokh. 2021. HeteroFL: Computation and communication efficient federated learning for heterogeneous clients. In *9th International Conference on Learning Representations*. OpenReview.net, Vienna, Austria. https://doi.org/10.48550/arXiv.2010.01264

[7] Moming Duan, Duo Liu, Xinyuan Ji, Renping Liu, Liang Liang, Xianzhang Chen, and Yujuan Tan. 2021. FedGroup: Efficient Federated Learning via Decomposed Similarity-Based Clustering. In *2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*. IEEE, New York City, NY, USA, 228–237. https://doi.org/10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00042

[8] Qi Feng, Debiao He, Jian Shen, Min Luo, and Kim-Kwang Raymond Choo. 2024. PpNNT: Multiparty Privacy-Preserving Neural Network Training System. *IEEE Transactions on Artificial Intelligence* 5, 1 (2024), 370–383. https://doi.org/10.1109/TAI.2023.3247554

[9] Jixiang Gan, Lei Zeng, Qi Liu, and Xiaodong Liu. 2023. A survey of intelligent load monitoring in IoT-enabled distributed smart grids. *International Journal of Ad Hoc and Ubiquitous Computing* 42, 1 (2023), 12–29. https://doi.org/10.1504/IJAHUC.2023.127781

[10] Honghao Gao, Binyang Qiu, Ye Wang, Si Yu, Yueshen Xu, and Xinheng Wang. 2023. TBDB: Token Bucket-Based Dynamic Batching for Resource Scheduling Supporting Neural Network Inference in Intelligent Consumer Electronics. *IEEE Transactions on Consumer Electronics* 70, 1 (2023), 1134–1144. https://doi.org/10.1109/TCE.2023.3339633

[11] Honghao Gao, Xuejie Wang, Wei Wei, Anwer Al-Dulaimi, and Yueshen Xu. 2024. Com-DDPG: Task Offloading Based on Multiagent Reinforcement Learning for Information-Communication-Enhanced Mobile Edge Computing in the Internet of Vehicles. *IEEE Transactions on Vehicular Technology* 73, 1 (2024), 348–361. https://doi.org/10.1109/TVT.2023.3309321

[12] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. 2022. An Efficient Framework for Clustered Federated Learning. *IEEE Transactions on Information Theory* 68, 12 (2022), 8076–8091. https://doi.org/10.1109/TIT.2022.3192506

[13] Filip Hanzely and Peter Richtárik. 2020. Federated learning of a mixture of global and local models. *arXiv preprint arXiv:2002.05516* (2020). https://doi.org/10.48550/arXiv.2002.05516

[14] Chaoyang He, Murali Annavaram, and Salman Avestimehr. 2020. Group knowledge transfer: Federated learning of large cnns at the edge. *Advances in Neural Information Processing Systems* 33 (2020), 14068–14080. https://proceedings.neurips.cc/paper_files/paper/2020/file/a1d4c20b182ad7137ab3606f0e3fc8a4-Paper.pdf

[15] Wenji He, Haipeng Yao, Fu Wang, Zunliang Wang, and Zehui Xiong. 2023. Enhancing the Efficiency of UAV Swarms Communication in 5G Networks through a Hybrid Split and Federated Learning Approach. In *2023 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE, Marrakesh, Morocco, 1371–1376. https://doi.org/10.1109/IWCMC58020.2023.10183145

[16] Yutao Huang, Lingyang Chu, Zirui Zhou, Lanjun Wang, Jiangchuan Liu, Jian Pei, and Yong Zhang. 2021. Personalized cross-silo federated learning on non-iid data. In *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI, Vancouver, Canada, 7865–7873. https://doi.org/10.1609/aaai.v35i9.16960

[17] Yili Jiang, Kuan Zhang, Yi Qian, and Liang Zhou. 2022. Anonymous and efficient authentication scheme for privacy-preserving distributed learning. *IEEE Transactions on Information Forensics and Security* 17 (2022), 2227–2240. https://doi.org/10.1109/TIFS.2022.3181848

[18] Latif U. Khan, Mohsen Guizani, Ala Al-Fuqaha, Choong Seon Hong, Dusit Niyato, and Zhu Han. 2024. A Joint Communication and Learning Framework for Hierarchical Split Federated Learning. *IEEE Internet of Things Journal* 11, 1 (2024), 268–282. https://doi.org/10.1109/JIOT.2023.3315673

[19] Safiullah Khan, Wai-Kong Lee, Angshuman Karmakar, Jose Maria Bermudo Mera, Abdul Majeed, and Seong Oun Hwang. 2023. Area–Time Efficient Implementation of NIST Lightweight Hash Functions Targeting IoT Applications. *IEEE Internet of Things Journal* 10, 9 (2023), 8083–8095. https://doi.org/10.1109/JIOT.2022.3229516

[20] Bo Li, Mikkel N. Schmidt, Tommy S. Alstrøm, and Sebastian U. Stich. 2023. On the Effectiveness of Partial Variance Reduction in Federated Learning with Heterogeneous Data. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Vancouver, Canada, 3964–3973. https://doi.org/10.1109/CVPR52729.2023.00386

[21] Peisong Li, Ziren Xiao, Xinheng Wang, Kaizhu Huang, Yi Huang, and Honghao Gao. 2024. EPtask: Deep Reinforcement Learning Based Energy-Efficient and Priority-Aware Task Scheduling for Dynamic Vehicular Edge Computing. *IEEE Transactions on Intelligent Vehicles* 9, 1 (2024), 1830–1846. https://doi.org/10.1109/TIV.2023.3321679

[22] Bingyan Liu, Yao Guo, and Xiangqun Chen. 2021. PFA: Privacy-preserving Federated Adaptation for Effective Model Personalization. In *Proceedings of the Web Conference 2021*. ACM, Ljubljana, Slovenia, 923–934. https://doi.org/10.1145/3442381.3449847

[23] Ji Liu, Jizhou Huang, Yang Zhou, Xuhong Li, Shilei Ji, Haoyi Xiong, and Dejing Dou. 2022. From distributed machine learning to federated learning: A survey. *Knowledge and Information Systems* 64, 4 (2022), 885–917. https://doi.org/10.1007/s10115-022-01664-x

[24] Jianchun Liu, Qingmin Zeng, Hongli Xu, Yang Xu, Zhiyuan Wang, and He Huang. 2023. Adaptive Block-Wise Regularization and Knowledge Distillation for Enhancing Federated Learning. *IEEE/ACM Transactions on Networking* 32, 1 (2023), 791–805. https://doi.org/10.1109/TNET.2023.3301972

[25] Qi Liu, Yuanyuan Jin, Xuefei Cao, Xiaodong Liu, Xiaokang Zhou, Yonghong Zhang, Xiaolong Xu, and Lianyong Qi. 2024. An Entity Ontology-Based Knowledge Graph Embedding Approach to News Credibility Assessment. *IEEE Transactions on Computational Social Systems* 11, 4 (2024), 5308–5318. https://doi.org/10.1109/TCSS.2023.3342873

[26] Qi Liu, Yang Li, Muhammad Bilal, Xiaodong Liu, Yonghong Zhang, Huihui Wang, Xiaolong Xu, and Hui Lu. 2023. CFNet: An Eigenvalue Preserved Approach to Multiscale Building Segmentation in High-Resolution Remote Sensing Images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 16 (2023), 2481–2491. https://doi.org/10.1109/JSTARS.2023.3244336

[27] Qi Liu, Zhiyun Yang, Ru Ji, Yonghong Zhang, Muhammad Bilal, Xiaodong Liu, S. Vimal, and Xiaolong Xu. 2023. Deep Vision in Analysis and Recognition of Radar Data: Achievements, Advancements, and Challenges. *IEEE Systems, Man, and Cybernetics Magazine* 9, 4 (2023), 4–12. https://doi.org/10.1109/MSMC.2022.3216943

[28] Qi Liu, Lei Zeng, Muhammad Bilal, Houbing Song, Xiaodong Liu, Yonghong Zhang, and Xuefei Cao. 2023. A Multi-Swarm PSO Approach to Large-Scale Task Scheduling in a Sustainable Supply Chain Datacenter. *IEEE Transactions on Green Communications and Networking* 7, 4 (2023), 1667–1677. https://doi.org/10.1109/TGCN.2023.3283509

[29] Yan Liu, Tian Wang, Shaoliang Peng, Guojun Wang, and Weijia Jia. 2021. Edge-Based Model Cleaning and Device Clustering in Federated Learning. *Chinese Journal of Computers* 44, 12 (2021), 2515–2528. http://cjc.ict.ac.cn/online/onlinepaper/liuy-20211217120533.pdf

[30] Bing Luo, Wenli Xiao, Shiqiang Wang, Jianwei Huang, and Leandros Tassiulas. 2022. Tackling system and statistical heterogeneity for federated learning with adaptive client sampling. In *IEEE INFOCOM 2022-IEEE conference on computer communications*. IEEE, London, UK, 1739–1748. https://doi.org/10.1109/INFOCOM48880.2022.9796935

[31] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. PMLR, Ft. Lauderdale, FL, USA, 1273–1282. https://proceedings.mlr.press/v54/mcmahan17a.html

[32] Truc Nguyen and My T. Thai. 2024. Preserving Privacy and Security in Federated Learning. *IEEE/ACM Transactions on Networking* 32, 1 (2024), 833–843. https://doi.org/10.1109/TNET.2023.3302016

[33] Kilian Pfeiffer, Martin Rapp, Ramin Khalili, and Jörg Henkel. 2023. Federated Learning for Computationally Constrained Heterogeneous Devices: A Survey. *Comput. Surveys* 55, 14s (July 2023), 1–27. https://doi.org/10.1145/3596907

[34] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, Salt Lake City, UT, USA, 4510–4520. https://doi.org/10.1109/CVPR.2018.00474

[35] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. 2021. Clustered Federated Learning: Model-Agnostic Distributed Multitask Optimization Under Privacy Constraints. *IEEE Transactions on Neural Networks and Learning Systems* 32, 8 (2021), 3710–3722. https://doi.org/10.1109/TNNLS.2020.3015958

[36] Reent Schlegel, Siddhartha Kumar, Eirik Rosnes, and Alexandre Graell i Amat. 2023. CodedPaddedFL and CodedSecAgg: Straggler Mitigation and Secure Aggregation in Federated Learning. *IEEE Transactions on Communications* 71, 4 (2023), 2013–2027. https://doi.org/10.1109/TCOMM.2023.3244243

[37] Jinglong Shen, Nan Cheng, Xiucheng Wang, Feng Lyu, Wenchao Xu, Zhi Liu, Khalid Aldubaikhy, and Xuemin Shen. 2024. RingSFL: An Adaptive Split Federated Learning Towards Taming Client Heterogeneity. *IEEE Transactions on Mobile Computing* 23, 5 (2024), 5462–5478. https://doi.org/10.1109/TMC.2023.3309633

[38] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. 2017. Federated multi-task learning. *Advances in neural information processing systems* 30 (2017), 4424–4434. https://proceedings.neurips.cc/paper_files/paper/2017/file/6211080fa89981f66b1a0c9d55c61d0f-Paper.pdf

[39] Mengkai Song, Zhibo Wang, Zhifei Zhang, Yang Song, Qian Wang, Ju Ren, and Hairong Qi. 2020. Analyzing user-level privacy attack against federated learning. *IEEE Journal on Selected Areas in Communications* 38, 10 (2020), 2430–2444. https://doi.org/10.1109/JSAC.2020.3000372

[40] Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. 2023. Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems* 34, 12 (2023), 9587–9603. https://doi.org/10.1109/TNNLS.2022.3160699

[41] Chandra Thapa, Pathum Chamikara Mahawaga Arachchige, Seyit Camtepe, and Lichao Sun. 2022. Splitfed: When federated learning meets split learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. AAAI, Virtual, 8485–8493. https://doi.org/10.1609/aaai.v36i8.20825

[42] Paul Voigt and Axel Von dem Bussche. 2017. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing* 10, 3152676 (2017), 10–5555. https://link.springer.com/content/pdf/bbm:978-3-319-57959-7/1 Publisher: Springer.

[43] Joel Wolfrath, Nikhil Sreekumar, Dhruv Kumar, Yuanli Wang, and Abhishek Chandra. 2022. HACCS: Heterogeneity-Aware Clustered Client Selection for Accelerated Federated Learning. In *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, Lyon, France, 985–995. https://doi.org/10.1109/IPDPS53621.2022.00100

[44] Hao Wu, Qi Liu, Xiaodong Liu, Yonghong Zhang, and Zhiyun Yang. 2022. An edge-assisted cloud framework using a residual concatenate FCN approach to beam correction in the internet of weather radars. *World Wide Web* 25, 5 (2022), 1923–1949. https://doi.org/10.1007/s11280-021-00988-y

[45] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv preprint arXiv:1708.07747* (2017). https://doi.org/10.48550/arXiv.1708.07747

[46] Ziyuan Yang, Yingyu Chen, Huijie Huangfu, Maosong Ran, Hui Wang, Xiaoxiao Li, and Yi Zhang. 2023. Dynamic Corrected Split Federated Learning with Homomorphic Encryption for U-shaped Medical Image Networks. *IEEE Journal of Biomedical and Health Informatics* 27, 12 (2023), 5946–5957. https://doi.org/10.1109/JBHI.2023.3317632

[47] Mang Ye, Xiuwen Fang, Bo Du, Pong C. Yuen, and Dacheng Tao. 2023. Heterogeneous Federated Learning: State-of-the-Art and Research Challenges. *Comput. Surveys* 56, 3 (Oct. 2023), 1–44. https://doi.org/10.1145/3625558

[48] Benshun Yin, Zhiyong Chen, and Meixia Tao. 2023. Predictive GAN-Powered Multi-Objective Optimization for Hybrid Federated Split Learning. *IEEE Transactions on Communications* 71, 8 (2023), 4544–4560. https://doi.org/10.1109/TCOMM.2023.3277878

[49] Jinliang Yuan, Shangguang Wang, Shihe Wang, Yuanchun Li, Xiao Ma, Ao Zhou, and Mengwei Xu. 2023. Privacy as a Resource in Differentially Private Federated Learning. In *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*. IEEE, New York City, NY, USA, 1–10. https://doi.org/10.1109/INFOCOM53939.2023.10228953

[50] Hongwei Zeng, Zhongzhi Zhu, Ye Wang, Zhengzhe Xiang, and Honghao Gao. 2024. Periodic Collaboration and Real-Time Dispatch Using an Actor–Critic Framework for UAV Movement in Mobile Edge Computing. *IEEE Internet of Things Journal* 11, 12 (2024), 21215–21226. https://doi.org/10.1109/JIOT.2024.3366506

[51] Lei Zeng, Qi Liu, Shigen Shen, and Xiaodong Liu. 2024. Improved Double Deep Q Network-Based Task Scheduling Algorithm in Edge Computing for Makespan Optimization. *Tsinghua Science and Technology* 29, 3 (2024), 806–817. https://doi.org/10.26599/TST.2023.9010058

[52] Songge Zhang, Haoyu Tu, Zuguang Li, Shengbo Liu, Shaofeng Li, Wen Wu, and Xuemin Sherman Shen. 2023. Cluster-HSFL: A Cluster-Based Hybrid Split and Federated Learning. In *2023 IEEE/CIC International Conference on Communications in China (ICCC)*. IEEE, Dalian, China, 1–2. https://doi.org/10.1109/ICCC57788.2023.10233408

[53] Jiali Zheng, Yixin Chen, and Qijia Lai. 2024. PPSFL: Privacy-Preserving Split Federated Learning for Heterogeneous Data in Edge-Based Internet of Things. *Future Generation Computer Systems* 156 (2024), 231–241. https://doi.org/10.1016/j.future.2024.03.020

[54] Xiaokang Zhou, Wang Huang, Wei Liang, Zheng Yan, Jianhua Ma, Yi Pan, and Kevin I.-Kai Wang. 2024. Federated distillation and blockchain empowered secure knowledge sharing for Internet of medical Things. *Information Sciences* 662 (2024), 120217. https://doi.org/10.1016/j.ins.2024.120217

[55] Xiaokang Zhou, Wei Liang, Akira Kawai, Kaoru Fueda, Jinhua She, and Kevin I-Kai Wang. 2024. Adaptive Segmentation Enhanced Asynchronous Federated Learning for Sustainable Intelligent Transportation Systems. *IEEE Transactions on Intelligent Transportation Systems* 25, 7 (2024), 6658–6666. https://doi.org/10.1109/TITS.2024.3362058

[56] Xiaokang Zhou, Wei Liang, Kevin I-Kai Wang, Zheng Yan, Laurence T. Yang, Wei Wei, Jianhua Ma, and Qun Jin. 2023. Decentralized P2P Federated Learning for Privacy-Preserving and Resilient Mobile Robotic Systems. *IEEE Wireless Communications* 30, 2 (2023), 82–89. https://doi.org/10.1109/MWC.004.2200381

[57] Xiaokang Zhou, Qiuyue Yang, Qiang Liu, Wei Liang, Kevin Wang, Zhi Liu, Jianhua Ma, and Qun Jin. 2024. Spatial–Temporal Federated Transfer Learning with multi-sensor data fusion for cooperative positioning. *Information Fusion* 105 (2024), 102182. https://doi.org/10.1016/j.inffus.2023.102182

[58] Xiaokang Zhou, Qiuyue Yang, Xuzhe Zheng, Wei Liang, Kevin I-Kai Wang, Jianhua Ma, Yi Pan, and Qun Jin. 2024. Personalized Federated Learning With Model-Contrastive Learning for Multi-Modal User Modeling in Human-Centric Metaverse. *IEEE Journal on Selected Areas in Communications* 42, 4 (2024), 817–831. https://doi.org/10.1109/JSAC.2023.3345431

[59] Xiaokang Zhou, Xiaozhou Ye, Kevin I-Kai Wang, Wei Liang, Nirmal Kumar C. Nair, Shohei Shimizu, Zheng Yan, and Qun Jin. 2023. Hierarchical Federated Learning With Social Context Clustering-Based Participant Selection for

Internet of Medical Things Applications. *IEEE Transactions on Computational Social Systems* 10, 4 (2023), 1742–1751. https://doi.org/10.1109/TCSS.2023.3259431

[60] Xiaokang Zhou, Xuzhe Zheng, Xuesong Cui, Jiashuai Shi, Wei Liang, Zheng Yan, Laurence T. Yang, Shohei Shimizu, and Kevin I-Kai Wang. 2023. Digital Twin Enhanced Federated Reinforcement Learning With Lightweight Knowledge Distillation in Mobile Networks. *IEEE Journal on Selected Areas in Communications* 41, 10 (2023), 3191–3211. https://doi.org/10.1109/JSAC.2023.3310046