Hierarchical Clustering-based Personalized Federated Learning for Robust and Fair Human Activity Recognition

YOUPENG LI, Xidian University, China
XUYU WANG*, Florida International University, United States
LINGLING AN*, Xidian University, China

Currently, federated learning (FL) can enable users to collaboratively train a global model while protecting the privacy of user data, which has been applied to human activity recognition (HAR) tasks. However, in real HAR scenarios, deploying an FL system needs to consider multiple aspects, including system accuracy, fairness, robustness, and scalability. Most existing FL frameworks aim to solve specific problems while ignoring other properties. In this paper, we propose FedCHAR, a personalized FL framework with a hierarchical clustering method for robust and fair HAR, which not only improves the accuracy and the fairness of model performance by exploiting the intrinsically similar relationship between users but also enhances the robustness of the system by identifying malicious nodes through clustering in attack scenarios. In addition, to enhance the scalability of FedCHAR, we also propose FedCHAR-DC, a scalable and adaptive FL framework which is featured by dynamic clustering and adapting to the addition of new users or the evolution of datasets for realistic FL-based HAR scenarios. We conduct extensive experiments to evaluate the performance of FedCHAR on seven datasets of different sizes. The results demonstrate that FedCHAR could obtain better performance on different datasets than the other five state-of-the-art methods in terms of accuracy, robustness, and fairness. We further validate that FedCHAR-DC exhibits satisfactory scalability on three large-scale datasets regardless of the number of participants.

CCS Concepts: • **Human-centered computing** → *Ubiquitous and mobile computing systems and tools.*

Additional Key Words and Phrases: Human activity recognition, federated learning, attack and defense, fairness

ACM Reference Format:
Youpeng Li, Xuyu Wang, and Lingling An. 2023. Hierarchical Clustering-based Personalized Federated Learning for Robust and Fair Human Activity Recognition. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 7, 1, Article 20 (March 2023), 38 pages. https://doi.org/10.1145/3580795

1 INTRODUCTION

Human activity recognition (HAR) can be widely used in several applications such as health monitoring, fall monitoring, smart home, augmented reality, and other scenarios[13][46][6]. With the rapid development of deep learning techniques, a large number of works have applied deep learning methods for HAR[5][7][28], but these works generally require the collection of user data for centralized training, which undoubtedly leaks the privacy of users (e.g., health data, locations).

*Corresponding author.

Authors' addresses: Youpeng Li, youpengl@stu.xidian.edu.cn, Guangzhou Institute of Technology, Xidian University, Xi'an, China; Xuyu Wang, xuywang@fiu.edu, Knight Foundation School of Computing and Information Sciences, Florida International University, Miami, Florida, United States; Lingling An, all@mail.xidian.edu.cn, School of Computer Science and Technology, Xidian University, Xi'an, China.

2474-9567/2023/3-ART20 $15.00
https://doi.org/10.1145/3580795

Federated learning (FL) is an emerging distributed machine learning method that can be used to address the problem of users' privacy leakage. For FL, users only need to upload their local models to the server for aggregation to share training results among users. In the classic federated averaging algorithm[30], the user first uploads the model parameters to the server, and the server uses the weighted average method to aggregate the user model parameters and sends the updated model to all participating users. The training generally iterates several communication rounds until the model converges. However, the direct application of FL to HAR scenarios will face three major challenges. First, data heterogeneity among users leads to unfair model performance due to the fact that a single global model cannot fit the data distribution of all users simultaneously. Second, the FL system lacks robustness. For example, the FL training process is vulnerable to poisoning attacks, leading to significant degradation of model performance. Third, the scalability of the FL system is insufficient; the FL system hardly adapts to unseen situations (e.g., new user participation or evolution of user activity data), resulting in the inability of the FL system to scale to large-scale user training scenarios.

Specifically, due to data heterogeneity among users, only training a single global model is insufficient to fit the data distribution of all users, resulting in unsatisfactory performance. For example, a small number of users have particular activity categories but most users do not have. The aggregated global model may only learn general features but fail to learn user-specific features. Moreover, the fairness in FL can be achieved if users have similar model accuracy. In the above case, the model performance for a small fraction of users is below average, leading to an unfair model performance among users in HAR. To handle the problem of data heterogeneity among users, different types of personalized FL methods have been proposed, such as clustering [44], federated transfer learning [53], meta-learning [20], and federated multi-task learning [41]. Although these methods can eliminate the influence of data heterogeneity among users to a certain extent, most methods are limited to specific scenarios and have poor generalization ability. For example, in clustering-based FL methods [44][4][12][37], users can be clustered by exploiting the inherent similarity of data distributions among users, and the impact among users with dissimilar data distributions is eliminated. However, they do not consider the attack scenario in the FL setting. For example, some malicious nodes upload poisoned models to the server, which significantly degrades the performance of the global model. In addition, the above approaches do not consider the addition of new users or data evolution of users in real scenarios of FL-based HAR, resulting in the poor generalization ability of the model.

Furthermore, in real HAR scenarios, deploying an FL system needs to consider multiple factors, including system accuracy, fairness, robustness, and scalability. Most existing methods aim to solve a specific problem but ignore other properties. For example, one way to improve fairness is to assign larger weights to user models with unsatisfactory performance so that the fairness of model performance among users can be improved [21]. However, this method is very susceptible to attack. If the poisoned model uploaded by the malicious node is incorrectly assigned a larger weight, it will have a negative impact on the training process. To improve the robustness, various robust aggregation methods are used to reduce the interference of malicious nodes [3][42][52]. However, it is easy to filter out valuable information during the aggregation process, leading to unsatisfactory and unfair model performance among users. In fact, it is challenging to design a general framework that considers all of the above properties. Considering the limitations of the above research, we focus on simultaneously improving accuracy, fairness, and robustness in HAR.

In this paper, we propose FedCHAR, a personalized **Fed**erated learning framework with a hierarchical **C**lustering-based method for robust and fair **HAR**, which not only improves the accuracy and the fairness of model performance by exploiting intrinsically similar relationships among users in the benign scenario but also improves the robustness of the system by identifying malicious nodes through clustering in the attack scenario (i.e., weakening the impact of the attack). Our method breaks the constraints between accuracy, fairness, and robustness and realizes fair and robust HAR in FL scenarios with different sensors such as inertial measurement unit (IMU), radar, and depth camera. In addition, to enhance the scalability of FedCHAR, we propose FedCHAR-DC, a scalable and adaptive FL framework which is featured by dynamic clustering and adapting to the addition of new users or the evolution of datasets for realistic FL-based HAR scenarios.

Our work is motivated by two key observations. First, the activity behaviors of different users may have inherently similar characteristics due to physiological characteristics (e.g., height, weight, etc.), environmental characteristics, and the same type of sensors. It motivates us to exploit the similarity between user data distributions and reduce the interference between dissimilar users through clustering. Second, different data distributions among users lead to different updating directions of user model parameters. Specifically, the similarity of users can be measured by calculating the cosine similarity of the updating directions of model parameters between users. Moreover, in the attack scenario, whether it is a label poisoning attack or model poisoning attack, the model parameters uploaded by malicious nodes are often different from those of benign users. It is natural to inspire us to consider that malicious nodes can be identified through clustering so that the training process of the models for benign users will not be influenced. Based on the above two findings, we introduce hierarchical clustering, which can create a hierarchy of clusters without knowing the fixed number of clusters. It not only enables multiple users with similar data distributions to share model parameters with each other, but also resists malicious attacks on the model aggregation.

We also evaluate the performance of FedCHAR using seven datasets collected by multiple types of sensors, including three large-scale and four small-scale datasets collected in different environments. Since these data are collected from different users, there is more or less statistical heterogeneity between the data distributions. Our evaluation results show that FedCHAR outperforms the other five state-of-the-art baselines in terms of accuracy while maintaining low variance (i.e., better fairness of model performance among users). In addition, through comparative experiments under the scenarios of four attack types, we find that only FedCHAR is immune to all types of attacks, indicating that FedCHAR has strong robustness to resist various types of attacks. Moreover, by combining dynamic clustering and a new user mechanism, FedCHAR-DC also demonstrates superior performance on three large-scale datasets of different sizes, which means that FedCHAR-DC could achieve satisfactory scalability in real FL-based HAR settings.

Our key contributions are summarized as follows:

- To the best of our knowledge, this is the first work that considers improving the accuracy, fairness, and robustness of the FL model simultaneously for the HAR problem. Also, our code is publicly available at github.com/youpengl/FedCHAR.
- We propose a personalized FL framework (i.e., FedCHAR) with a hierarchical clustering for robust and fair HAR, which not only improves the accuracy and the fairness of model performance by exploiting intrinsically similar relationships among users but also improves the robustness of the system by identifying malicious nodes through clustering in attack scenarios.
- We also design a scalable and adaptive FL framework (i.e., FedCHAR-DC) for large-scale users' training. Compared with previous works based on clustered FL for HAR, which assume that all users are participating in the clustering stage, FedCHAR-DC is featured by dynamic clustering and adapting to the addition of new users or the evolution of datasets for realistic FL-based HAR scenarios.
- We implement various comparative experiments to evaluate the performance of FedCHAR on seven datasets of different sizes. Compared with the other five state-of-the-art baselines in both benign and attack scenarios, FedCHAR shows superior performance in terms of accuracy, robustness, and fairness on seven datasets with different modalities. We further validate that FedCHAR-DC exhibits satisfactory scalability on three large-scale datasets regardless of the number of participants.

The rest of the paper is organized as follows. Section 2 and Section 3 introduce recent work and present our motivation for designing FedCHAR, respectively. Section 4 and 5 discuss the system framework and the problem formulation in detail, respectively. Section 6 introduces the experimental setup. Section 7 evaluates the performance of FedCHAR. In Section 8, we discuss the limitations of this work and future works. In Section 9, we conclude this paper.

2 RELATED WORK

2.1 Human Activity Recognition

HAR can be used for many types of applications (e.g., locomotion [35], daily activity recognition [25], and gesture recognition [34]). Currently, different sensors and wireless techniques have been exploited for HAR applications, including WiFi [38], RFID [36], and wearable sensors (e.g., acclerometer[9], gyroscope [8], acoustic sensor [47]). Traditional machine learning methods for HAR rely on manually extracting features, which requires expert knowledge and complex processing procedures [1][11]. Benefiting from automatic feature extraction, deep learning has been widely used in HAR tasks [13][5][7][28]. Most of them require high-quality labeled data, which is generally centralized in the cloud for training. However, the centralized data collection method leaks the user's data privacy information, such as health status and location.

2.2 Federated Learning

FL can effectively solve the problem of privacy leakage of user data [23][18][27]. However, for FL, devices in the network frequently generate and collect data in a non-independent and identically distributed (non-IID) fashion, and the number of data samples and label types collected by different devices are unbalanced. Due to the heterogeneity of user data, the trained global model using traditional FL algorithms (e.g., FedAvg[30]) often cannot fit the data of all users well. Our work is related to personalized FL, as well as its robustness and fairness as follows.

2.2.1 Personalized Federated Learning. To mitigate the impact of user data heterogeneity, personalized FL methods are proposed, including clustering-based FL [12][4][39][44][37], meta-learning[20], federated multi-task learning[41][29], and federated transfer learning[53]. For example, Ghosh et al. proposed a framework named Iterative Federated Clustering Algorithm (IFCA), whose limitation is that the server needs to send k models with k clusters to users, which will significantly increase communication overhead [12]. Sattler et al. proposed clustered FL, which recursively bisects users in a top-down fashion [39]. Briggs et al. proposed an intermediate clustering-based FL algorithm [4]. Tu et al. proposed FedDL, which merges users' local models with a bottom-up dynamic sharing scheme [44]. Ouyang et al. proposed ClusterFL, which uses the alternating direction method of multipliers (ADMM) algorithm to alternately update the clustering structure and model weights [37]. For personalized multi-task learning, Smith et al. proposed a primal-dual multi-task learning framework, which focuses on solving convex optimization problems[41]. Marfoq et al. proposed to study federated multi-task learning under the assumption that each user's data distribution is composed of an unknown mixture of distributions[29].

Although the above methods could reduce the impact of user data heterogeneity, none of them consider the situation where malicious nodes are involved in the training process. Ditto [22] considers attack scenarios and mitigates the impact of the attack by constraining the extent to which the user-personalized model references the global model. However, it will lead to the training of the user-personalized model towards local training. In real HAR scenarios, locally trained models often fail to achieve satisfactory test accuracy as the sample size among users is often small and unbalanced. Moreover, Ditto is only robust against limited types of attacks and is less robust to inner-product manipulation attacks. In addition, most of the above-mentioned clustering-based works fail to consider the scalability of FL systems and the generalization capability of the models. For example, the method proposed by briggs [4] requires all users to participate in the clustering stage. Even in ClusterFL [37] and FedDL [44], the authors assume that all users participate in all communication rounds, which fails to meet the scalability needs of FL systems in real HAR scenarios. Meanwhile, the above methods do not consider the addition of new users or user data evolution, which often occurs in real FL-based HAR scenarios, resulting in the poor generalization ability of the global model.

In this paper, FedCHAR not only exploits the similarity relationship of users through clustering, which further enables the trained model to better fit the data distribution of users through intra-cluster personalization, but also isolates some malicious nodes from benign nodes to improve the robustness of the FL system. In addition, benefiting from the proposed dynamic clustering and new user mechanism, FedCHAR-DC adapts to the new user addition and user data evolution in real HAR scenarios.

2.2.2 Robustness in Federated Learning. The FL framework greatly improves the model training accuracy and guarantees users' privacy. However, several attack methods could perturb the training process, mainly including data poisoning attacks and model poisoning attacks. For example, Bhagoji et al.[2] explored a range of attack strategies where the boosting model parameters uploaded by malicious devices dominate the training process. Xu et al.[50] also introduced an attack called Free-riders, which manipulates user devices to upload gradients randomly drawn from a uniform distribution. In addition, Tolpegin et al.[43] explored targeted data poisoning attacks in which a malicious device sends the model trained on mislabeled data to the server, thus disrupting the training process.

On the other hand, some defense approaches are proposed to improve the robustness of the FL system. Generally, robust aggregation methods can be used to reduce the impact of attacks (e.g., Krum[3], Multi-Krum[3], Bullyan[31], Trimmed Mean[52], Gradient Clipping[42], and Median[52]). However, these methods may filter out valuable information during the aggregation process, resulting in unsatisfactory and unfair model performance among users.

2.2.3 Fairness in Federated Learning. Due to data heterogeneity among users, the trained global model cannot effectively learn the personalized characteristics of all users, which easily leads to unfair model performance among users. Currently, few works focus on fair FL algorithms. Mohri et al.[32] proposed a fair FL framework where the centralized model is optimized for a mixture of distributions. Li et al.[21] proposed a reweighting approach to improve the fairness of model performance among users. Lee et al.[15] proposed to use zero-shot data augmentation to reduce the impact of heterogeneous user data and improve the fairness of model performance among users. However, the methods mentioned above are susceptible to attack. Once a larger weight is given to the poisoned model uploaded by the malicious device, it will have a negative impact on training.

The methods mentioned above three subsections are only suitable for solving specific problems and ignore other properties. In real HAR scenarios, deploying an FL system often needs to consider multiple aspects, such as accuracy, fairness, robustness, and scalability. FedCHAR breaks the constraints between the properties mentioned above (e.g., robustness and fairness), which not only improves the accuracy and the fairness of model performance by exploring intrinsically similar relationships among users in the benign scenario but also improves the robustness of the system by identifying malicious nodes through clustering.

3 MOTIVATION

Considering the challenges of data heterogeneity among users and vulnerability to poisoning attacks during global model training in FL-based HAR, we will analyze and discuss the inherent similarity of data distribution among users and the differences in optimization paths among users models, which motivates our proposed clustering-based FedCHAR in the following two subsections.

3.1 Similarity of Data Distributions among Users

The activity behaviors of different users may have inherently similar characteristics due to physiological characteristics (e.g., height, weight, etc.), environmental characteristics, and similar sensors. Exploiting the similarity of

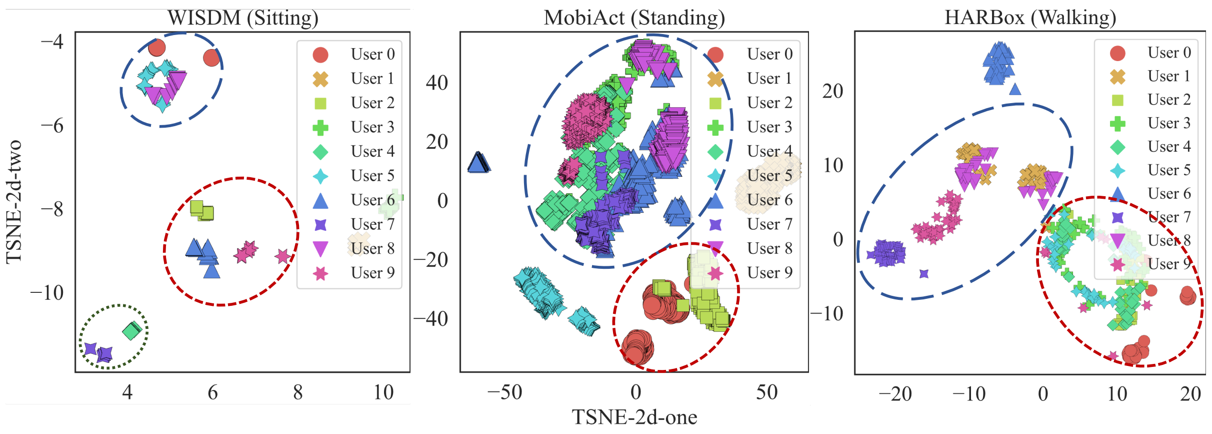
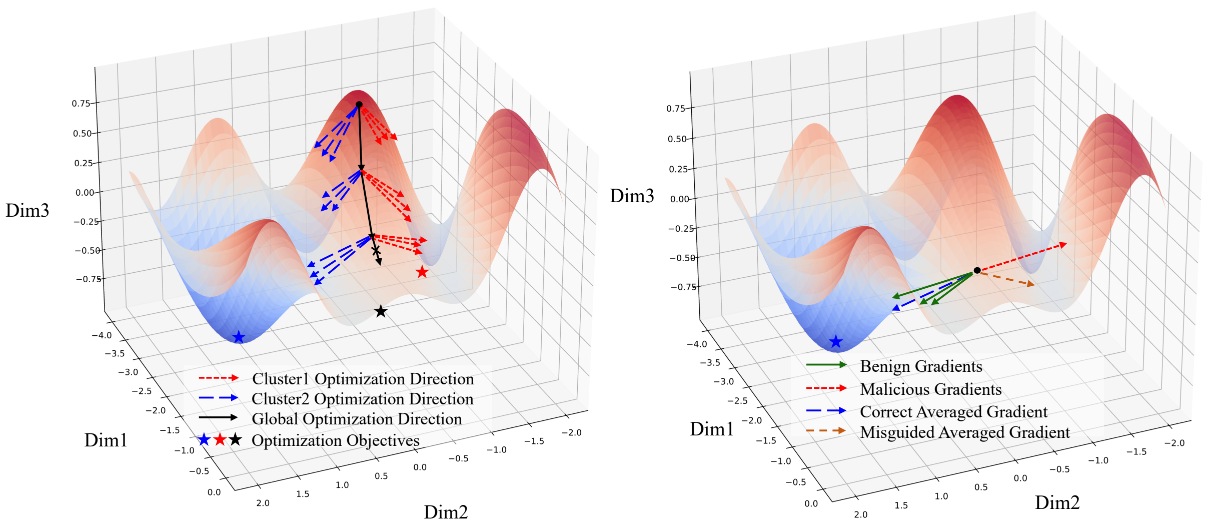


Fig. 1. Visualization of the user data distributions of three datasets after reducing the dimension to 2D using t-SNE.

data distributions among users can facilitate the collaboration among similar users. To visualize the data distributions of users, we select ten users in each of three datasets (i.e., WISDM, MobiAct, and HARBox) and present their data distribution with the corresponding categories (i.e., Sitting, Standing, and Walking). Since each sample in the dataset contains hundreds of dimensional features, we use t-SNE to reduce the feature dimensionality to 2-dimensions (2D). As shown in Fig. 1, each dataset has a clear clustering relationship in the data distributions among users.

3.2 Differences in Optimization Paths among Users Models



Dim3
Dim1
Dim2
Cluster1 Optimization Direction
Cluster2 Optimization Direction
Global Optimization Direction
Optimization Objectives
Benign Gradients
Malicious Gradients
Correct Averaged Gradient
Misguided Averaged Gradient

Fig. 2. The optimization direction of the model parameters of different users in each communication round.

Due to data heterogeneity among users, the updating direction of models among users is often inconsistent [54]. Users with a larger number of samples have more batches participating in the training of the local model, and more model updates mean more steps toward the objective of the local model. In addition, in the classical FedAvg algorithm [30], users with a large number of samples will be assigned greater weights in the global model aggregation. The larger the weights are, the more the updating direction of the global model is dominated by them, which will be detrimental to users with a small sample size to participate in collaborative training. The accuracy of their final models is much lower than average or even worse than local training, resulting in unfair model performance among users.

The left side of Fig. 2 shows the optimization direction of the model parameters of different users in each communication round. We can notice that as the number of communication rounds increases, the updating directions of the model parameters of users in different clusters will become more and more opposite (i.e., the cosine value of the model update parameters of two users in different clusters will become from positive to negative) [39]. Therefore, training with a few global rounds will make it easier to distinguish users with different data distributions. Inspired by the above finding, a natural idea is that if we can perform clustering at an appropriate point, users with similar updating directions can be more accurately clustered into the same cluster. Otherwise, they are assigned to different clusters, thus eliminating the influence of each other. In each cluster, the updated grouping model [1] after aggregating the model updates of similar users will better fit the data distribution of the users. Therefore, the poor global model fitting effect caused by non-IID problems can be solved, so the fairness of model performance among users is guaranteed.

Moreover, in attack scenarios, as shown on the right side of Fig. 2, when malicious attackers perform gradient ascent during local training or upload opposite model updates, the faulty model updates after aggregation will mislead the update of the global model, resulting in the accuracy degradation of benign user models. Inspired by the fact that the updating direction of the model parameters of malicious nodes is often different from that of benign nodes[2], it is natural to consider that malicious nodes can be isolated from benign nodes through clustering.

In summary, we find that clustering has a bifold benefit, as it not only clusters similar users into the same clusters, thus weakening the interference of dissimilar users, but also isolates malicious nodes from benign nodes to mitigate the impact of attacks. The two findings motivate us to design a clustering-based FL framework in this paper.

4 SYSTEM OVERVIEW

Fig. 3 shows the overall system architecture of FedCHAR. Specifically, we divide the communication rounds into three stages, including the initial communication rounds, the clustering stage, and the remaining communication rounds after clustering. In our FedCHAR framework, each user needs to train two models per round alternately. One is the user-personalized model, and the other is the global model or the grouping model corresponding to the initial or remaining rounds. The initial rounds consist of three steps. First, randomly selected users upload model updates to the server. Second, the server aggregates the model updates uploaded by users. Last, the server sends the updated global model to the users.

After the initial communication rounds, the server clusters the users by calculating the cosine similarity between the model update parameters of the users, which can reflect the similarity of data distributions between users. The server aggregates the model update parameters of users in the same cluster according to the clustering result and then sends the grouping model to the corresponding users. After clustering, only users in the same cluster can share parameters in the remaining communication rounds. In addition, for adapting to dynamic variations of user activities, the clustering stage can be repeated periodically (e.g., daily) using recently collected data.

[1]In this paper, in order to distinguish the global model used in the non-clustering FL algorithm, we refer to the global model of each cluster after clustering as the grouping model.

[2]In our context, user, node, device, and client have similar meanings. To be more contextual, we use different words in different positions.

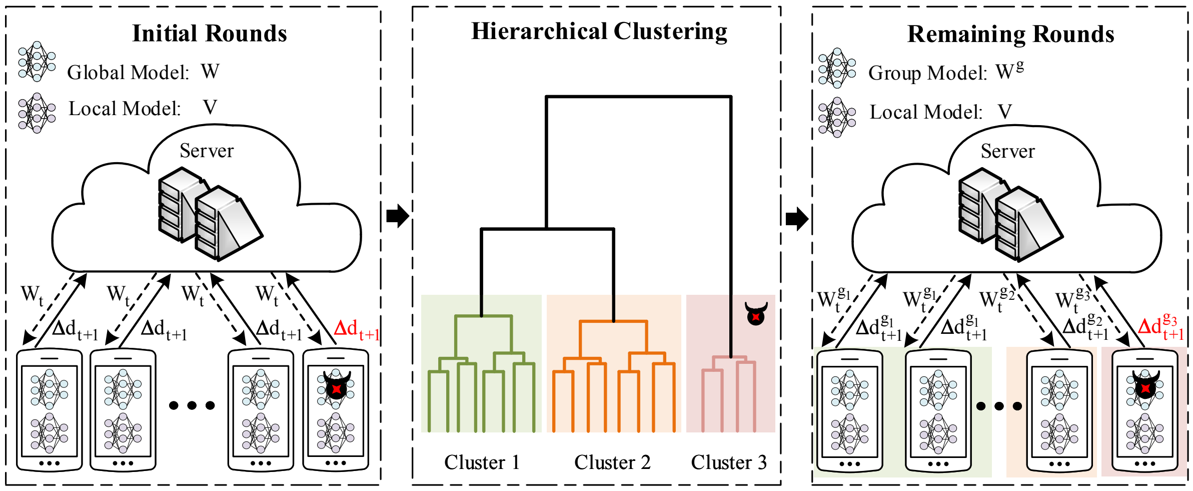


Fig. 3. Overview of the proposed FedCHAR.

5 SYSTEM MODEL

In this section, our FedCHAR system focuses on robust and fair HAR in the personalized FL framework. First, we discuss the definitions of robustness and fairness. Then, we discuss the FedCHAR model in detail.

5.1 Definitions of Robustness and Fairness

Our work mainly explores the robustness and fairness of FedCHAR, defined in detail below.

Robustness: To explore the robustness of FedCHAR, we introduce four types of attacks, one of which is label-based poisoning attack (i.e., A1) and the other three are model-based poisoning attacks (i.e., A2, A3, and A4) [22][49]. We report the average test accuracy among benign users under attack scenarios to measure robustness. Specifically, we consider model $w1$ to be more robust than model $w2$ if the average test accuracy among benign users using model $w1$ is higher than that among benign users using model $w2$ under attack scenarios. Furthermore, to explore the upper bound of FedCHAR's ability to resist attacks, following previous work [22] [3], we consider different attack ratios (i.e., 20% and 50%) in different experiments, respectively.

- **(A1) Label Poisoning:** Malicious nodes send model updates trained on samples with randomly scrambled labels to the server.
- **(A2) Random Updates:** Malicious nodes calculate the standard deviation *std* of the model update parameters, randomly generate Gaussian distribution model update parameters, and then send them to the server.
- **(A3) Model Replacement:** Malicious nodes amplify the model update parameters several times and upload them to the server for the model aggregation. In this work, our default magnification is ten times.
- **(A4) Inner Product Manipulation:** Malicious nodes upload the inverse of the model update parameters to the server so that the inner product between the faulty aggregated model updates and the correct model updates is negative, thus making the optimization path of the global model deviate from the correct optimization path.

Fairness: In the FL setting, the definition of fairness is often related to characteristics such as non-IID data among users, client selection, and model heterogeneity. Our work focuses on the unfairness issue caused by the non-IID problem of data among users. Specifically, affected by factors (e.g., environment, equipment, and individual differences), the distribution of HAR data collected from different users is often heterogeneous, and there are various forms of distribution skew. Moreover, due to different economic conditions and equipment, the amount of data collected varies by user. According to the different living habits of different users, the number of each category in the user sample is also unbalanced. The above series of problems will lead to the occurrence of the unfairness problem. For example, when the number of user samples is inconsistent, users with a large number of samples will have more contribution or offset to the training of the global model, resulting in the update of the global model being dominated by users with a large number of samples. In term of the heterogeneity of user data in HAR, the global model obtained by FL training may fit some users' data (with common features) well, which is measured in the high accuracy of the user model. However, the global model does not fit the data of the other users (with more unique features), and the user model accuracy is lower than average or even worse than local training.

Based on the above analysis, this paper defines fairness as the uniformity of model performance distribution among users (existing FL papers have adopted the same definition of fairness [32][22][24][48][16][17]), which is more suitable for FL-based HAR scenarios with heterogeneous data among users. We measure fairness by calculating the variance of the accuracy among benign devices. Specifically, we consider model $w1$ to be fairer than model $w2$ if the variance of the accuracy among benign users using model $w1$ is smaller than that among benign users using model $w2$.

5.2 Problem Formulation

We assume a cross-device FL scenario for HAR, where K devices are connected to a central server and each device $k \in [K]$ has its own local dataset S_k with $|S_k| = m_k$ data samples. The dataset S_k is obtained from the local data distribution D_k. Under the standard FL framework, each device shares the model trained by local data to obtain an optimal global model, which ideally fits all user data. The FL model aims to solve the following problem,

$$\min_{w} G(F_1(w), \cdots, F_K(w)), \qquad \text{(Global Objective)}$$

where G is an aggregation function. In FedAvg [30], G is actually a weighted average function, that is, $G = \sum_{k=1}^{K} p_k F_k(w)$, where we denote $p_k = m_k / \sum_{k=1}^{K} m_k$, and $F_k(w)$ represents the user local objective for user k, which is defined by,

$$F_k(w) = \frac{1}{|S_k|} \sum_{x \in S_k} f(w, x), \qquad \text{(User Local Objective)}$$

where $f(w, x)$ is the loss function for device k.

However, in the real scene of FL-based HAR, the data distribution among users is non-IID due to users' individual differences. There are many cases where the data distribution among users deviates from the IID distribution [18]. As shown in Table 1, we describe in detail the common cases of non-IID in FL in the context of the characteristics of HAR data. In the real FL-based HAR scenario, the data between users is often composed of a mixture of the above types of distribution skew. How to solve a series of problems caused by the non-IID of the data between users (e.g., making unsatisfactory and unfair global model performance) is a difficult challenge.

Due to the heterogeneity of users' data, the obtained global model fails to learn the personalized features of all users, resulting in poor model performance. Moreover, if the user only uses the local dataset for training, a large number of local data samples are required to represent the actual data distribution. However, because of the limitations of the user's equipment and environmental factors, the sample size is often small and insufficient to represent the actual data distribution, resulting in model overfitting.

Table 1. Various distribution skews that cause non-IID of data among users.

In the following formula x represents the feature and y represents the label.	
Type	**Description**
Feature Distribution Skew	For different users, their $P(x)$ is different, even if $P(y\|x)$ is the same. For example, in the HAR dataset, different users have different feature distributions when performing the same activity due to individual differences (e.g., action range and speed).
Label Distribution Skew	Different users have different $P(y)$ even though $P(x\|y)$ is the same. In a more realistic HAR scenario, users have different numbers of samples from different classes. For example, user A is a young person who walks or runs more often, and user B is an elderly person who lies or sits more often. Even some users have samples for categories that most users do not have, such as falls that are more prone to occur in older adults.
Concept Drift	Different users have different $P(x\|y)$ even though $P(y)$ is the same. The same label corresponds to very different features under different conditions (e.g. environment, time). For example, the Depth dataset used in the evaluation experiments in this paper is collected in dark and bright outdoor locations.
Concept Shift	For different users their $P(y\|x)$ is different even if $P(x)$ is the same. For example, in a FL-based finger gesture recognition scenario involving people from multiple countries, the same finger gesture corresponds to different or even opposite meanings in different countries, which will cause users in different countries to mark the same finger gesture for different labels.
Quantity Skew	This is the most common and closest to reality scenario. Affected by factors such as equipment and environment, the number of collected user data samples is different, which may also lead to label distribution skew.

5.3 FedCHAR Objective

Motivated by federated multi-task learning, we explore a hierarchical clustering-based personalized FL framework for robust and fair HAR to address the above problems. Assume that clusters already exist among users. In each cluster, we consider two tasks: (i) optimizing the grouping objective $G_g(\cdot)$ for updating grouping model w_g. (ii) optimizing the user local objective $F_k(v_k)$ for updating user-personalized model v_k. To link these two tasks, we introduce a regularization coefficient λ to control the distance between the user-personalized model v_k and grouping model w_g. In each cluster, the optimization problem for each device $k \in [C]$ is formulated by,

$$\begin{aligned} \min_{v_k} u_k(v_k; w_g) &:= F_k(v_k) + \frac{\lambda}{2}\|v_k - w_g\|^2 \\ s.t. \quad w_g &\in \arg\min_{w} G_g(F_1(w), \cdots, F_C(w)), \end{aligned} \tag{FedCHAR Objective}$$

where $G_g(\cdot)$ represents the grouping objective for cluster g, C represents the number of users in cluster g, and the regularization coefficient λ controls the distance between user-personalized model v_k and grouping model w_g. When λ is equal to 0, FedCHAR objective is reduced to the user local objective $F_k(v_k)$ (i.e., local training). The larger value of λ, the closer v_k is to w_g. When λ is ∞, FedCHAR objective is approximately equal to the grouping objective $G_g(\cdot)$ (i.e., global FL training).

5.4 Algorithm for Solving FedCHAR Objective

To solve the above FedCHAR Objective, we propose to update the grouping models $\{w_g\}_{(g\in G)}$ and the user-personalized models $\{v_k\}_{(k\in K)}$ jointly in each round with an alternating manner.

Algorithm 1 FedCHAR

```
Input: $K, T_0, T_1, I, P, \sigma, \lambda, \eta, w_0, \{v_0^k\}_{k\in[K]}$
Output: $\{v^k\}_{k\in[K]}$
 1: Initialize $w_0$, $\{v_0^k\}_{k\in[K]}$
 2: $w_{T_0} \leftarrow FederatedMultiTaskLearning(w_0, T_0, K)$                                  ▷ Initial Rounds
 3: $\{w_{T_0}^k\}_{k\in[K]} \leftarrow w_{T_0}$ // Server sends $w_{T_0}$ to all $K$ devices.
 4: for all device k in parallel do                                                               ▷ Clustering Stage
 5:     $w_{T_0+1}^k = w_{T_0}^k - \eta\nabla F_k(w_{T_0}^k)$ // Device $k$ runs $I$ epochs updating global model $w_{T_0}^k$.
 6:     $\triangle d_{T_0+1}^k = w_{T_0+1}^k - w_{T_0}^k$ // Device $k$ calculates $\triangle d_{T_0+1}^k$ and sent it to server for clustering.
 7: end for
 8: $G \leftarrow AgglomerativeHierarchicalClustering(\{\triangle d_{T_0+1}^k\}, \sigma)$
 9: for cluster $g \in G$ in parallel do                                                           ▷ Remaining Rounds
10:     $w_0^g \leftarrow w_{T_0}$
11:     $w_{T_1}^g \leftarrow FederatedMultiTaskLearning(w_0^g, T_1, |g|)$
12: end for
13: function $FederatedMultiTaskLearning(w_t, T, N)$
14:     for $t = 0, \cdots, T-1$ do
15:         // Server randomly selects a subset $S_t$ among $N$ devices then sends model $w_t$ to selected devices.
16:         $|S_t| = P \times N$
17:         $\{w_t^k\}_{k\in S_t} \leftarrow w_t$
18:         for device $k \in S_t$ in parallel do
19:             $v_{t+1}^k = v_t^k - \eta(\nabla F_k(v_t^k) + \lambda(v_t^k - w_t^k))$ // Device $k$ runs $I$ epochs updating $w_t^k$ and $v_t^k$.
20:             $w_{t+1}^k = w_t^k - \eta\nabla F_k(w_t^k)$
21:             $\triangle d_{t+1}^k = w_{t+1}^k - w_t^k$
22:         end for
23:         $w_{t+1} \leftarrow ServerAggregate(w_t, \{\triangle d_{t+1}^k\})$
24:     end for
25: end function
```

Below we will describe the entire algorithm flow in detail. We divide the proposed Algorithm 1 into three modules, namely the initial rounds, the clustering stage, and the remaining rounds after clustering.

The initial rounds include T_0 communication rounds in total. Each communication round consists of the following two steps:

(1) In the t round, the server randomly selects a subset S_t from all devices, and sends the global model w_t to the selected devices. Each device (e.g., device k) runs minibatch stochastic gradient descent (SGD) for I epochs locally to obtain the global model w_{t+1}^k and the user-personalized model v_{t+1}^k. It is noted that the updating order of the global model and the user-personalized model can be exchanged. Then, device k calculates the model update parameter $\triangle d_{t+1}^k$ ($\triangle d_{t+1}^k = w_{t+1}^k - w_t$) and sends it to the server.
(2) The server aggregates the model update parameters of all participating users to obtain the global model w_{t+1} for the next communication round.

After the initial rounds end, the global model w_{T_0} obtained by the server in the latest aggregation is sent to all K devices. Subsequently, each device locally runs minibatch SGD for I epochs to update the global model, and sends the model update parameters to the server for clustering.

In the clustering stage, we use an agglomerative hierarchical clustering method, which is a greedy algorithm that starts with a cluster for each user and ends with all users in one cluster [33]. Specifically, each user belongs to a different cluster in the initial stage (i.e., K users are divided into K clusters). Then server calculates the cosine similarity of the model update parameters between the devices, which can be expressed by,

$$simi(i,j) := \frac{\langle \triangle d^i, \triangle d^j \rangle}{\|\triangle d^i\| \cdot \|\triangle d^j\|} \quad (i, j \in S),$$

where $simi(i,j)$ represents the cosine similarity between the model update parameters of user i and the model update parameters of user j, and its value range is $[-1, 1]$. The closer the value is to -1, the more opposite the updating directions of the user model i and user model j are, and the closer the value is to 1, the more similar the updating directions of the user model i and user model j are. S is the set of participating users in the current stage.

Then, the server chooses a pair of users with the most similar updating directions of model to cluster into a group, and the total number of clusters is $K-1$ at this time. In each subsequent clustering, the server calculates the distance (i.e., the cosine similarity) between two different clusters, and then merges the two clusters with the smallest distance into a new cluster. Three common linkage criteria are used to measure the distance between clusters, namely complete linkage, single linkage, and average linkage [51]. In addition, there are some efficient algorithms for linkage methods that can reduce the time complexity to $O(n^2)$ or even lower, such as the SLINK algorithm [40] for single linkage and the CLINK algorithm [10] for complete linkage. We will compare the performance of above three linkage criteria on different datasets in Section 7.7.

Generally, the agglomerative hierarchical clustering method starts with a cluster for each user and end with all users in one cluster. If there are K users, the server will cluster $K-1$ times during the entire clustering process. The server can analyze the intermediate results produced by the entire clustering process and select an appropriate clustering threshold to determine the number of clusters.

The last stage is the remaining rounds. The server will notify all devices that the clustering stage is completed. After all devices receive the notification, they will use w_{T_0} as their initial grouping model. The remaining rounds include T_1 communication rounds. For each cluster (e.g., the cluster g), each subsequent communication round includes the following steps.

(1) In the t round, the server randomly selects a subset S_t from the cluster g. Then the server sends the grouping model w_t^g to the devices belonging to the cluster g. Each device (e.g., device k) runs minibatch SGD for I epochs locally to obtain the grouping model $w_{t+1}^k (k \in g)$ and the user-personalized model v_{t+1}^k. The updating order of the grouping model and the user-personalized model can be also exchanged. Finally, device k calculates the model update parameter $\triangle d_{t+1}^k$ ($\triangle d_{t+1}^k = w_{t+1}^k - w_t^g$) and sends it to the server.
(2) The server aggregates the model update parameters of the devices in the cluster g to obtain the new grouping model w_{t+1}^g for the next communication round.

When all communication rounds are over, all devices will use the user-personalized models for HAR. The proposed FedCHAR algorithm can effectively improve the robustness and fairness for HAR with two reasons. First, some malicious nodes can be automatically separated by the agglomerative hierarchical clustering method, which greatly reduces the interference caused by the tampered model updates from malicious nodes during the training process, thus improving the robustness of the FL model. Second, users with similar data distributions will be clustered into the same cluster, which can mitigate the impact of heterogeneous user data, thus enhancing the fairness of model performance among users.

In addition, we also discuss the computation and communication overhead of the proposed FedCHAR method. During the initial communication rounds, the user side only has the computation overhead when training locally and computing the model update parameters. In this work, the number of user training epochs on all datasets does not exceed 5, which greatly saves the computational overhead on the user side. Moreover, uploading the model update parameters to the server after the user calculates the model update parameters is more secure than uploading the model parameters directly. The server side only has the computational overhead when aggregating users' model update parameters, the size of which is determined by the number of participating users. For communication overhead, only one model is delivered between the server and each user in each communication round. In the clustering stage, the computational overhead of the server is mainly used to calculate the distance between different clusters using the agglomerative hierarchical clustering method. The server can select a corresponding efficient algorithm according to different linkage criteria to calculate the distance between different clusters, and control the time complexity to $O(n^2)$ or lower. During the remaining communication rounds after clustering, since each cluster can complete the training in parallel, the computational overhead and communication overhead are similar to the initial communication rounds.

Through the above analysis, we conclude that FedCHAR has no additional and excessive communication overhead and computational overhead compared to the traditional FL algorithm. Moreover, in the field of FL, the computational overhead of the user side is more important because of the limited computing power of the user equipment. In our FedCHAR work, the agglomerative hierarchical clustering method used is performed on the server side.

5.5 FedCHAR with Dynamic Clustering for Large-scale FL Scenario

In this section, we provide an in-depth analysis of the challenges (i.e., scalability and adaptability of FL systems) encountered in more realistic FL-based HAR scenarios, which have motivated our proposal for FedCHAR-DC.

In FL-based cross-devices HAR, where the total population size is in the thousands [18], deploying an FL system with scalability is necessary. However, scalability seems to be bounded by clustering. Intuitively, in clustering-based FL algorithms, the more users participate in the clustering stage, the better the clustering effect. Although satisfactory system performance can be achieved in extant clustering-based FL paradigms, they often require all users to participate in the clustering stage or even the entire FL training process [37][4]. However, this assumption is too strong and does not fit the real cross-device scenario. Moreover, when clustering is performed on the server side, the similarity of all users needs to be calculated, which undoubtedly increases the computational overhead of the system on the server side and is not conducive to large-scale users for training. Therefore, it is challenging to perform clustering without affecting the scalability of the FL system in the cross-device scenario.

In addition, in the HAR scenario, the data collected by user devices changes all the time. When users' behavioral habits change, their data distribution will also change, leading to possible changes in the clusters to which users belong. Apart from this, the clustering structure needs to be updated as a steady stream of new users is added during FL training. In [4], although the authors implemented one-shot clustering, the clustering structure formed by clustering is constant throughout the whole FL training process. It can lead to the fact that the grouping model obtained by intra-cluster aggregation may be affected by the non-IID of the user data within the cluster when the distribution of users' data changes. Therefore, it is challenging to design an adaptive FL system that accommodates new users and changes in user data distribution.

5.6 Algorithm for FedCHAR-DC

To enhance the scalability of FedCHAR, we propose a scalable and adaptive FL framework FedCHAR-DC, which is featured by dynamic clustering and adapting to the new users joining or datasets evolving for realistic FL-based HAR scenarios. Compared to the FedCHAR algorithm, we extend the clustering and remaining rounds, and the initial rounds are consistent with Algorithm 1. Below we will describe the main changes relative to FedCHAR; see Algorithm 2 for more details.

Algorithm 2 FedCHAR-DC

```
Input: $K, T_0, T_1, I, P, R, \sigma, \lambda, \eta, w_0, \{v_0^k\}_{k\in[K]}$
Output: $\{v^k\}_{k\in[K]}$
1: Initialize $w_0$, $\{v_0^k\}_{k\in[K]}$
2: $w_{T_0} \leftarrow FederatedMultiTaskLearning(w_0, T_0, K)$                         ▷ Initial Rounds
3: $\{\triangle d_{T_0+1}^k\} \leftarrow FederatedMultiTaskLearning(w_{T_0}, 1, K)$                ▷ Clustering Stage
4: $G \leftarrow AgglomerativeHierarchicalClustering(\{\triangle d_{T_0+1}^k\}, \sigma)$
5: // Server aggregates model updates $\{\triangle d_{T_0+1}^k\}$ based on clustering structure $G$.
6: for $g \in G$ do
7:     $w_0^g \leftarrow ServerAggregate(w_{T_0}, \{\triangle d_{T_0+1}^k\}_{k\in g})$ // Initial grouping model.
8:     $\triangle d_0^g \leftarrow Average(\{\triangle d_{T_0+1}^k\}_{k\in g})$ // Initial averaged grouping model update.
9: end for
10: // Initialize the $state$ vector, containing the cumulative rounds that users have not been selected in a row.
11: $state = np.zeros(K)$
12: $\{w_{T_1}^g\}_{g\in G} \leftarrow FederatedMultiTaskLearning(\{w_0^g\}_{g\in G}, T_1, K)$          ▷ Remaining Rounds
13: function $FederatedMultiTaskLearning(\{w_t^g\}, T, N)$       ▷ Before remaining rounds, $\{w_t^g\} \iff w_t$
14:     for $t = 0, \cdots, T-1$ do
15:         // Server randomly selects a subset $S_t$ among $N$ devices.
16:         for $device\ k \in S_t\ in\ parallel$ do
17:             if before remaining rounds or (device $k$ once belonged to a cluster and $state[k] < R$) then
18:                 $w_t^k \leftarrow w_t^g$ // Server sends the model $w_t^g$ of the cluster to which device $k$ belong.
19:                 $v_{t+1}^k = v_t^k - \eta(\nabla F_k(v_t^k) + \lambda(v_t^k - w_t^k))$ // Device $k$ runs $I$ epochs updating $w_t^k$ and $v_t^k$.
20:                 $w_{t+1}^k = w_t^k - \eta \nabla F_k(w_t^k)$
21:                 $\triangle d_{t+1}^k = w_{t+1}^k - w_t^k$
22:             else
23:                 $\triangle d_{t+1}^k, \{\triangle d_0^g\}, \{w_t^g\} \leftarrow NewUserMechanism(w_{T_0}, \{\triangle d_0^g\}, \{w_t^g\})$
24:             end if
25:         end for
26:         // Server aggregates model updates $\{\triangle d_{t+1}^k\}$ based on the clustering structure $G$
27:         for $g \in G$ do
28:             $w_{t+1}^g \leftarrow ServerAggregate(w_t^g, \{\triangle d_{t+1}^k\}_{k\in g})$
29:         end for
30:     end for
31: end function
```

- **Clustering Stage:** Unlike FedCHAR, which requires all users to participate in the clustering stage, FedCHAR-DC only needs to select a subset of all users to participate in clustering. When the participating users update the global model w_{T_0} and upload model updates $\{\triangle d_{T_0}^k\}$ to the server, the server performs agglomerative hierarchical clustering to obtain the clustering structure G. According to the clustering structure G, the server aggregates within each cluster to obtain initial grouping models $\{w_0^g\}_{g\in G}$, calculates and saves the initial averaged grouping model updates$\{\triangle d_0^g\}_{g\in G}$ of all clusters, which are used for assigning clusters to new users in the subsequent rounds.

- **Remaining Rounds:** Before the remaining rounds begin, the server will initialize an all-zero vector *state* of length K, the elements of which represent the cumulative rounds that users have not been selected in a row. During each remaining round, the server first determines whether device k already has a cluster to which it belongs and whether the number of consecutive rounds of non-participation in training is smaller than R. If the above conditions are satisfied, the user is considered to belong to the previous cluster. The server will send the grouping model w_t^g to device k ($k \in g$). Device k updates the grouping model w_t^g and the user-personalized model v_t^k in an alternating fashion and returns model update $\triangle d_{t+1}^k$ to the server. The server aggregates the grouping model updates $\{\triangle d_{t+1}^k\}$ from participating devices to obtain the grouping model w_{t+1}^g for the next round. Otherwise, the server triggers the new user mechanism to determine the cluster to which the user belongs. The new user mechanism will be described in detail in the next section. In particular, the above consecutive rounds R determine whether the server needs to reallocate the cluster for device k, and its value depends on the actual situation. For example, in the FL-based HAR scenario, if the user's data distribution changes rapidly, the value of R should be appropriately small; otherwise, the value of R can be appropriately large.

Algorithm 3 New User Mechanism

```
Input: $w_{T_0}$, $\{\triangle d_0^g\}$, $\{w_t^g\}$
Output: $\triangle d_t^k$, $\{\triangle d_0^g\}$, $\{w_t^g\}$
1: $w_t^k \leftarrow w_{T_0}$ // Server sends $w_{T_0}$ to new user $k$.
2: $w_{t+1}^k = w_t^k - \eta \nabla F_k(w_t^k)$ // Device $k$ runs $I$ epochs updating model $w_t^k$.
3: $\triangle d_{t+1}^k = w_{t+1}^k - w_t^k$ // Device $k$ calculates $\triangle d_{t+1}^k$ then send it to server.
4: $min \leftarrow min\_simi(\{\triangle d_0^g\}_{g \in G})$ // Server calculates the **minimum** similarity among $\{\triangle d_0^g\}_{g \in G}$.
5: $max \leftarrow max\_simi(\{\triangle d_0^g\}_{g \in G}, \triangle d_{t+1}^k)$ // The **maximum** cosine similarity between $\triangle d_{t+1}^k$ and $\{\triangle d_0^g\}$.
6: if $max < min$ then
7:     $g\_belong = len(\{\triangle d_0^g\})$ // Assign a new cluster to device $k$.
8:     $\{w_t^g\}.append(w_{T_0})$ // Update grouping models.
9:     $\{\triangle d_0^g\}.append(\triangle d_{t+1}^k)$ // Update initial averaged grouping model updates.
10: end if
11: $w_t^k \leftarrow w_t^g$ // Server sends the grouping model $w_t^g$ of the cluster to which new user $k$ belongs.
12: $v_{t+1}^k = v_t^k - \eta(\nabla F_k(v_t^k) + \lambda(v_t^k - w_t^k))$ // Device $k$ runs $I$ epochs updating $w_t^k$ and $v_t^k$.
13: $w_{t+1}^k = w_t^k - \eta \nabla F_k(w_t^k)$
14: $\triangle d_{t+1}^k = w_{t+1}^k - w_t^k$ // Device $k$ calculates $\triangle d_{t+1}^k$, and then send it to server.
```

5.7 Algorithm for New User Mechanism

Since new users may join in FL training, we propose a new user mechanism (see Algorithm 3) to adapt this case. Specifically, the server sends the global model w_{T_0} obtained after initial rounds to the new user k, and the new user k updates the global model w_{T_0} and returns the model update $\triangle d_{t+1}^k$ to the server. The server calculates the similarity between the initial averaged grouping model updates $\{\triangle d_0^g\}_{g \in G}$ and obtains the minimum value *min*, which can be used as a threshold to determine whether a user should be assigned to a new cluster or not. Then the server calculates the similarity between the model update $\triangle d_{t+1}^k$ of the new user k and the initial averaged grouping model updates $\{\triangle d_0^g\}_{g \in G}$ and obtains the maximum value *max*.

Then the server records the clustering identity g_{belong} that is most similar to the new user k. Specifically, if *max* is greater than or equal to *min*, the server sends the grouping model w_t^g of the cluster g_{belong} to the new

Table 2. Seven datasets of different sizes collected by multiple sensing devices in different environments.

Application	**Task**	**(# Subject, # Sample, # Feature)**	**Sensor**	**Environment**
Walking Activity Recognition (IMU)	Walking on -Corridors -Upstairs -Downstairs	(7, 1369, 900)	LPMS-B2 IMU	Node0,1,2,3 in Building1 Node4,5,6 in Building2
Human Movement Detection (UWB)	With/Without Human Movement	(8, 663, 55)	Decawave DWM1000 UWB	Node0,1 in Parking Lot Node2,3,4 in Corridor Node5,6,7 in Room
Gesture Recognition (FMCW)	Push/Pull Slide Left/Slide Right Clockwise Turning Counterwise Turning	(9, 10650, 54272)	TI AWR1843 mmWave Radar	Node0,1,2 in Room1 Node2,3,4 in Room2 Node5,6,7 in Room3
Gesture Recognition (Depth Camera)	Good/Ok Victory/Stop/Fist	(9, 7422, 1296)	PicoZense DCAM710	Node0,1,2 in Outdoor Node3,4,5 in Dark Node6,7,8 in Indoor
WISDM: ADL Recognition (Smartphone)	Walking/Jogging Sitting/Standing Upstairs/Downstairs	(36, 5471, 600)	Accelerometer	Controlled Laboratory Conditions with Sampling Rate: 20Hz
MobiAct: ADL Recognition (Smartphone)	Standing/Walking Jogging/Jumping Upstairs/Downstairs Stand to Sit	(57, 38633, 600)	Accelerometer and Gyroscope	57 Subjects(20-47y) Sampling Rate: 20Hz
HARBox: ADL Recognition (Smartphone)	Walking/Hopping Phone Calls Waving/Typing	(121, 32935, 900)	77 Different Smartphone Models	121 Subjects(17-55y) Sampling Rate: 43.5-57.5Hz

user k and updates the clustering structure G. Then the new user k updates the grouping model w_t^g and the user-personalized model v_t^k in an alternate manner. If *max* is less than *min*, the new user k is not similar to all the current clusters. The server will allocate a new cluster for the new user k and update the clustering structure G, and then use the global model w_{T_0} obtained after the initial rounds as the initial grouping model of the newly generated cluster and send it to the new user k. Then the server updates the grouping model set $\{w_t^g\}_{g\in G}$ and the initial averaged grouping model updates set $\{\triangle d_0^g\}_{g\in G}$. Meanwhile, the new user k updates the grouping model w_t^g and the user-personalized model v_t^k in an alternating fashion and returns the model update $\triangle d_{t+1}^k$ to the server.

6 EXPERIMENT SETUP

In Section 6.1, we describe the details of the seven datasets involved in the experiments. In Section 6.2, we introduce the experimental settings from four aspects including the details of data partitioning (see Section 6.2.1), the selection of hyperparameters (see Section 6.2.2), the models used (see Section 6.2.3), and the experimental platform (see Section 6.2.4). In Section 6.3, we describe the six baselines used in the experiments. In Section 6.4, we introduce various robust aggregation methods. In Section 6.5, we introduce three linkage criteria.

6.1 Datasets

In our experiment, we evaluate the performance of the proposed framework FedCHAR using seven public datasets of different sizes. They cover a variety of sensor device types with different activity recognition tasks in different environments. Moreover, these datasets are suitable for the FL settings because they are collected among different users, devices, and environments with significant dynamics. Moreover, different-sized datasets can help us estimate the scalability of FedCHAR.

6.1.1 Walking Activity Recognition with IMU[37]. The IMU dataset consists of seven participants (four males and three females) using an Inertial Measurement Unit (IMU) module to perform three walking activities (i.e., walking in the corridor, walking upstairs, and waking downstairs) in two buildings. The sampling rate of the IMU is 50Hz, and the chosen time window is 2s, where each sample contains 900-dimensional features. The detailed description of the IMU dataset is shown in Table 2.

6.1.2 Human Movement Detection with UWB Radar[37]. Radar is an active sensing system that uses radio waves to determine objects' distance, angle, and radial velocity. Since radar is not sensitive to light and weather conditions, it is more suitable for HAR tasks than visual cameras. In addition, it does not require any sensing equipment to be placed on the user, which brings portability and protects the user's privacy. To detect if there are human movements in a specific area, two Ultra Wide Band (UWB) radars spaced 3m apart are deployed in three different environments (i.e., parking lot, corridor, and room) for human movement detection. This dataset contains eight subjects, each of which walks randomly in the area for 10 minutes. The two-way ranging with the sample rate 5Hz is captured and labeled manually. Each data sample comprises 10 seconds of records (50 dimensions) and five statistical attributes (5 dimensions). The detailed description of the UWB dataset is shown in Table 2.

6.1.3 Gesture Recognition with FMCW Radar[26]. This is an open-source mmWave gesture dataset collected from various domains (i.e., environments, users, and locations). It can be used to develop domain-independent gesture recognition systems based on Frequency Modulated Continuous Wave (FMCW) radar. Since the total size of the processed dataset is too large, we choose the gesture data collected from nine volunteers, three environments, and five locations. Each environment has different sizes and furniture placement, resulting in different multipath effects. Therefore, the environment setup is more challenging for gesture recognition. The six gestures include push, pull, swipe left, swipe right, turn clockwise and turn counterclockwise. Each location has a different distance and angle from the radar, ranging from 0.6m to 1m and -30° to 30°. Each volunteer is asked to perform each gesture five times in each position. Volunteers are told how to perform each gesture prior to data collection and to keep other body parts relatively still while performing the gesture. The gesture duration range is from 0.5s to 2s. We normalize the values of the obtained dynamic range angle image (DRAI) sequences to $[0, 1]$. Since each DRAI sequence has a different length, we uniformly resize the sequence to 53×32×32 pixels using zero padding. A detailed description of the mmWave gesture dataset is shown in Table 2.

6.1.4 Gesture Recognition with Depth Camera[37]. Unlike conventional cameras, depth cameras protect users' privacy to a certain extent and have been widely used in gesture recognition and activity monitoring. The depth dataset consists of nine subjects performing five gestures (i.e., good, ok, victory, stop, and fist) in three environments (i.e., outdoor, dark, and indoor) using the depth camera. After obtaining the depth gesture's region of interest (ROI), the depth value is normalized to $[0, 1]$, and the depth image is reshaped to 36×36 pixels. The detailed description of the Depth dataset is shown in Table 2.

6.1.5 Activity of Daily Life (ADL) Recognition with Smartphones: WISDM[19]. The latest version of the WISDM dataset is obtained by collecting activity information from 36 subjects recorded by the accelerometer of an

Android smartphone. Following previous work [19], we choose a sampling frequency of 20Hz and a window size of 10s. Each data sample contains 600-dimensional features.

6.1.6 Activity of Daily Life (ADL) Recognition with Smartphones: MobiAct[45]. The latest version of the MobiAct dataset is obtained by collecting activity information from 66 subjects recorded by the accelerometer, gyroscope, and orientation sensors of a Samsung Galaxy S3 smartphone. For uniformity and completeness of sampling, we eliminate some users with missing or incomplete data and select 57 subjects. Following previous work [45], we choose a sampling frequency of 20Hz, a window size of 5s, and an overlap ratio of 80%. Each data sample contains features (600 dimensions) collected by accelerometers and gyroscopes.

6.1.7 Activity of Daily Life (ADL) Recognition with Smartphones: HARBox[37]. Smart devices (e.g., smartphones) are widely used to monitor daily human activity because of their built-in sensing units, including accelerometers, gyroscopes, and magnetometers. The HARBox dataset is collected by 121 users using an Android program with 77 smartphones. The sampling rate is also 50Hz, the sliding window is 2s, and each sample has 900-dimensional features. The detailed description of the HARBox dataset is shown in Table 2.

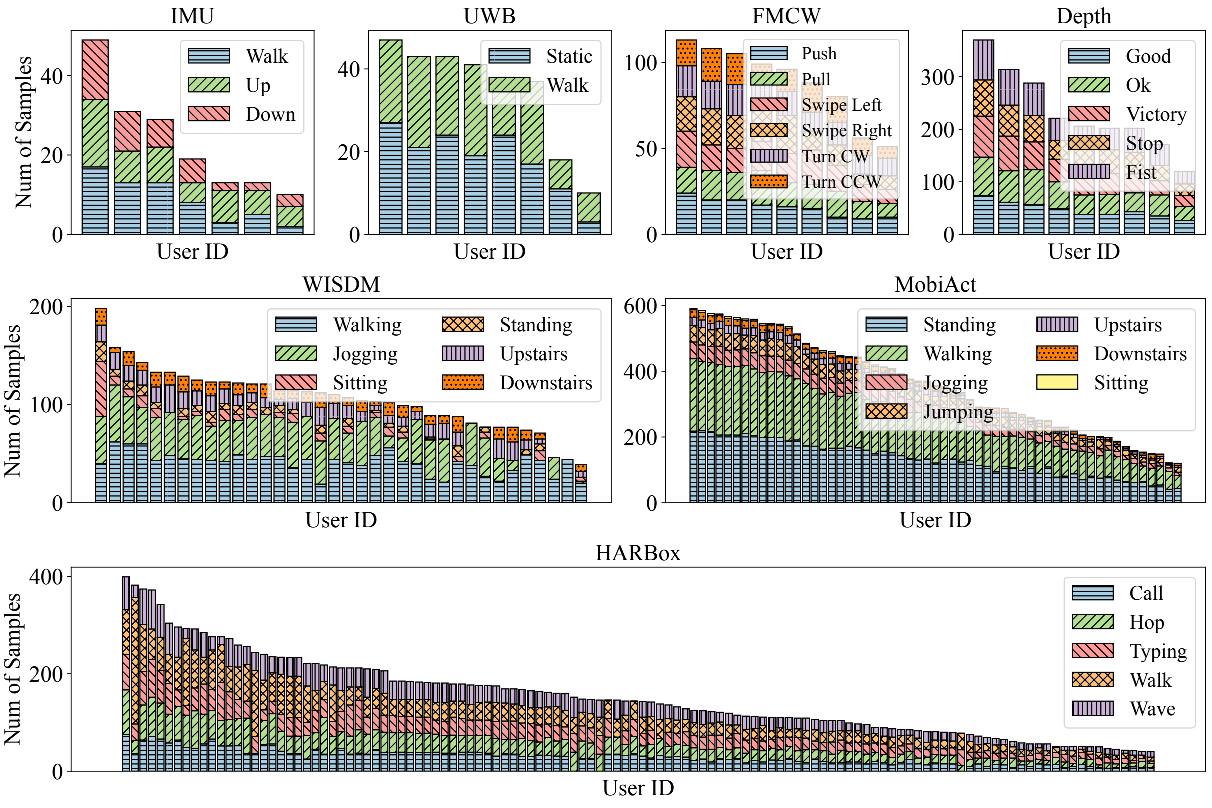


Fig. 4. Visualization of the data distribution of different users in the seven datasets. We can clearly find several characteristics: (1) the number of samples owned by different users is unbalanced; (2) the number of label types owned by different users is unbalanced (especially in large-scale datasets).

6.2 Setup

6.2.1 Data Partitioning. In order to simulate the user data distribution in FL-based HAR real-world scenarios, we implement different processing on different datasets. For the IMU dataset, the user dataset is divided into the training set, validation set, and test set with a ratio of [0.54, 0.06, 0.4]. Each user randomly selects 10-50 training samples from the training set for training. For the UWB dataset, the user dataset is divided in the ratio of [0.63, 0.07, 0.3], and each user randomly selects 10-50 training samples from the training set. For the FMCW dataset, each user randomly selects 50-150 samples from the dataset, and the user dataset is divided in the ratio of [0.72, 0.08, 0.2]. For the Depth dataset, we adopt the original partition method of the dataset [37]. Each user randomly selects 100-400 training samples from the training set and 10% from the training set as the validation set. For the WISDM dataset, since the amount of data among users in the original dataset is already unbalanced, we keep the original form. The user dataset is divided in the ratio of [0.63, 0.07, 0.3]. For the MobiAct dataset, the user dataset is divided in the ratio of [0.72, 0.08, 0.2], and each user randomly selects 100-600 training samples from the training set. For the HARBox dataset, each user randomly selects 50-500 samples from the dataset, and the user dataset is divided in the ratio of [0.72, 0.08, 0.2]. We report the results of all experiments on the test set.

Such random extraction and division methods are inspired by the insufficient number of user data samples for training and the imbalance of data samples among users in real FL scenarios. As shown in Fig. 4, we visualize the distribution owned by different users. We find that there are significant differences in the data distribution of each user, mainly in the unbalanced number of samples for each user, resulting in the different number of samples for different label types (i.e., different $P(y)$) in each user dataset. In addition, there are also differences in the activity characteristics for different users (i.e., different $P(x)$).

6.2.2 Hyperparameters. We conduct a large number of hyperparameter searches to finally determine the learning rate of 0.01 for IMU, UWB, FMCW, WISDM, and MobiAct datasets and 0.001 for Depth and HARBox datasets. The batch size for IMU and UWB datasets is 5, and the batch size for FMCW and Depth is 16. For the WISDM, MobiAct, and HARBox datasets, the batch size is 32. For the communication rounds, a total of 100 communication rounds are trained for the HARBox dataset. For the other datasets, the total number of global rounds is 50. For the number of local training epochs, for Depth and HARBox datasets, users have trained five epochs per round, and for other datasets, users have trained two epochs per round. For the number of participants per round, for small-scale datasets, IMU, UWB, FMCW, and Depth, all users participate in each round by default. For large-scale datasets, WISDM, MobiAct, and HARBox, half of the users participate in each round by default. In addition, we conduct a large number of experiments and discuss the selection of various hyperparameters in the experimental section.

6.2.3 Model Setup. For the model configuration, we use a neural network with one fully connected layer of 300 units for IMU-based and HARBox-based activity recognition and one fully connected layer of 16 units for UWB-based human movement detection. For depth camera-based gesture recognition, we choose a custom convolutional neural network with two convolutional layers with max pooling and kernel size 5×5, the number of filters of 2 convolutional layers from 8 to 16, and 2 fully connected layers of units [300, 100]. For FMCW-based gesture recognition, we design a neural network consisting of a frame model which employs a convolutional neural network (CNN) to extract spatial features from each single DRAI and a sequence model which utilizes long short-term memory (LSTM) to learn the temporal dependencies of the entire DRAI sequence. Specifically, the frame model has two convolution layers with max pooling and kernel size 5×5, one fully connected layer with 128 units, and one dropout layer with a rate of 0.5. The filters of the two convolutional layers increase from 8 to 16. The sequence model consists of 1 LSTM layer with a size of 128 hidden units and one fully connected layer with 128 inputs. We use neural networks for the WISDM-based and MobiAct-based activity recognition networks by stacking two layers of LSTM with 30 neurons in each layer. Network structures for FMCW, WISDM, and MobiAct are trained with Adam optimizer, and others are trained with SGD optimizer.

6.2.4 Platform. All experiments are conducted with 2 NVIDIA 3080TI GPUS. The PC (Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz 56-core) runs Ubuntu 20.04. The algorithms are implemented in TensorFlow.

6.3 Baselines

In our experiment, we compare the performance of FedCHAR with six baselines as follows.

6.3.1 Ditto[22]. Ditto is a personalized federated multi-task learning framework. It considers two tasks: the global objective $G(F_1(w), \cdots, F_K(w))$, and the user's local objective $F_k(v_k)$. To relate two tasks, a regularization coefficient λ is introduced to encourage the personalized models to be close to the global model. Specifically, each selected device k solves $\min_{v_k} h_k(v_k; w^*) := F_k(v_k) + \frac{\lambda}{2}\|v_k - w^*\|^2$, where $w^* \in \arg\min_{w} G(F_1(w), \cdots, F_K(w))$.

6.3.2 FedAvg[30]. FedAvg is a federated averaging algorithm. The server aggregates model parameters uploaded by users using a weighted average strategy. Specifically, in each communication round, parts of users are randomly selected for training, where each selected device k solves $\min_{w} F_k(w)$, where $F_k(w) = \sum_{i \in P_k} f_i(w)/n_k$, $f_i(w)$ represents the loss function, n_k represents the number of user training samples, and P_k represents the set of the user training sample. Then the server uses the weighted average method to aggregate the model parameters uploaded by all participating users and returns the updated global model to the users. The above process iterates over multiple communication rounds until the global model converges.

6.3.3 Fine-tuning[22]. Fine-tuning is a federated transfer learning method. Unlike Ditto, which jointly updates the personalized model and global model. The Fine-tuning method first obtains the global model w^* through several communication rounds of training, and then the server sends the global model w^* to all participating users. All participating users leverage the local dataset to fine-tune the global model. Specifically, each participating user optimizes the objective $h_k(v_k; w^*)$, where $\min_{v_k} h_k(v_k; w^*) := F_k(v_k) + \frac{\lambda}{2}\|v_k - w^*\|^2$, s.t. $w^* \in \arg\min_{w} G(F_1(w), \cdots, F_K(w))$.

6.3.4 ClusterFL[37]. ClusterFL is a clustering-based federated multi-task learning framework that simultaneously updates the global model and explores similar relationships among users. Specifically, the global objective is $\min_{W,C} G(F_1(w), \cdots, F_K(w)) + \alpha tr(WW^T) - \beta tr(F^T WW^T F)$, where the first term is the sum of empirical errors among all nodes, the second and third terms can be written as $(\alpha - \beta) \sum_{i=1}^{K} \|w_i\|_2^2 + \beta \sum_{j=1}^{M} \sum_{v \in C_j} \|w_v - \overline{w_j}\|_2^2$, where K represents the number of all nodes, M represents the number of clusters, and C represents the clustering structure. The above equation contains two regularization terms. The first regularization term is used to prevent overfitting of the user models. The second is introduced to make the user model w_v approach the average of all user models.

6.3.5 L2SGD[14]. L2SGD formulates a new optimization of FL, which regularizes personalized models towards their mean. However, the method requires all devices to participate in each round of communication. In order to maintain consistency with other baselines, we follow previous work and adopt a different solver under the premise of the same objective to allow parts of users to participate in each round of communication [22]. Specifically, each selected device k solves $\min_{w} F_k(w) + \frac{\lambda}{2}\|w - \overline{w}\|^2$, where $\overline{w} = \frac{1}{N}\sum_{k=1}^{N} w_k$ is the mean of model weights of participants.

6.3.6 Local Training. Each device trains the local model using its own dataset.

6.4 Robust Aggregation Methods

6.4.1 Krum[3]. Krum selects the one most similar to the other models among the n local models as the global model. The intuition is that even if the model of a malicious node is selected, it has a high degree of similarity to the models of other benign nodes. Thus, its influence will be limited.

Specifically, it is assumed that there are n user devices, where m are malicious nodes. First, the user model is flattened into a vector V, and the distance matrix D is obtained by calculating the Euclidean distance between different user model vectors. The ith row in the distance matrix D represents the distance between the user model vector V_i and other user model vectors $V_j(j \in [1, n], j \neq i)$. Then, we sort each row of the distance matrix D from small to large and add the first $n - m - 2$ values to obtain the score s. Finally, the user model with the smallest score is selected as the updated global model.

In this work, we set m as the product of the proportion of malicious devices in all devices and the number of participating devices in each round. For example, if there are n user devices, the proportion of malicious devices is α, and the number of participating user devices in the t round is n_t, then $m = \alpha \times n_t$.

6.4.2 Multi-krum[3]. Multi-Krum is an extended version of Krum. The only difference from Krum is that Multi-Krum is the average user model parameters corresponding to the first k minimum scores. When k =1, Multi-Krum is reduced to Krum. When $k = n$, Multi-Krum is equivalent to the averaging method. In this work, it is assumed that there are n user devices, and the number of user devices participating in the t round is n_t, and then $k = n_t - m$.

6.4.3 Clipping[42]. The clipping method can defend against model parameter amplification attacks (i.e., A3). Because the model parameter amplification attack will increase the norm of the model update parameters, once the value exceeds the threshold set by the server, the server will proportionally reduce the model update parameters to resist the impact of attacks or noise.

Specifically, we assume that the set of users randomly selected in the t round is S_t. After the users upload the local model parameters to the server, the server will calculate the L2 norm of the model update parameters before aggregation. If this value exceeds a previously set threshold, the server will scale down the model update parameters. The aggregation process can be expressed by,

$$\triangle d_{t+1} = \sum_{k \in S_t} \frac{\triangle d_{t+1}^k}{\max(1, \| \triangle d_{t+1}^k \|_2 / M)},$$

where M is the threshold set by the server. In this work, we set the threshold M as the median of the L2 norm of all user model update parameters.

6.4.4 K-norm. K-norm calculates the L2 norm of the model update parameters uploaded by the users participating in the current round and then sorts them in ascending order. Finally, the mean of the first k numbers is the model update parameters. The definition of k is the same as that of k in Multi-Krum above.

6.4.5 Median[52]. The aggregation model consists of the median of the model parameters of each layer for all participating users. For example, assuming there are n users, the kth layer of the aggregation model is obtained by n users first flattening their respective kth layer parameters into a one-dimensional vector and then taking their medians.

6.5 Linkage Criteria

6.5.1 Average Linkage. Average-linkage clustering is divided into unweighted and weighted average linkage clustering. In this work, we adopt the first clustering method. In unweighted average linkage clustering, the distance between clusters equals the average of all distances between the two points belonging to the two clusters, respectively. Mathematically, the unweighted average linkage function can be described by the following expression: $D(A,B) = \sum_{a\in A}\sum_{b\in B}\frac{d(a,b)}{|A|\cdot|B|}$, where A and B represent two clusters, respectively, a is a user belonging to cluster A, and b is a user belonging to cluster B. $|A|$ and $|B|$ represent the number of users in cluster A and the number of users in cluster B, respectively.

6.5.2 Complete Linkage. Complete-linkage clustering is also known as farthest-neighbor clustering. In complete-linkage clustering, the link of two clusters consists of all the element pairs in the two clusters. The distance between clusters is equal to the farthest distance between the two points belonging to the two clusters respectively. Mathematically, the complete linkage function can be described by the following expression: $D(A,B) = \max_{a\in A,b\in B} d(a,b)$, where A and B represent two clusters, a is a user belonging to cluster A, and b is a user belonging to cluster B.

6.5.3 Single Linkage. Single-linkage clustering is also known as nearest-neighbor clustering. In single-linkage clustering, the link of two clusters consists of all the element pairs in the two clusters. The distance between clusters is equal to the shortest distance between the two points belonging to the two clusters, respectively. Mathematically, the single linkage function can be described by the following expression: $D(A,B) = \min_{a\in A,b\in B} d(a,b)$, where A and B represent two clusters, a is a user belonging to cluster A, and b is a user belonging to cluster B.

7 EVALUATION

In this section, we conduct extensive experiments to evaluate the performance of FedCHAR on seven datasets of different sizes. Specifically, we evaluate the performance of FedCHAR in terms of accuracy, fairness, and robustness of FedCHAR from various aspects, including the comparison with different state-of-the-art baselines.

In all experiments, we report the mean and variance of test accuracy among all benign users, which can measure robustness and fairness. By default, the values in the table represent the average test accuracy of all benign user models, with the variance of the test accuracies among all benign user models represented in brackets.

7.1 Comparison of Different Baselines

In this section, we evaluate the performance of FedCHAR by comparing it with different baselines. We use the barplot with error bars to show the performance of six baselines (i.e., Ditto, FedAvg, Fine-tuning, ClusterFL, L2SGD, and Local) on seven datasets (i.e., IMU, UWB, FMCW, Depth, WISDM, MobiAct, and HARBox) under benign and attack scenarios, respectively. We assume that malicious nodes account for 50% of the total number of nodes. The reason for using barplot with error bars is that it not only shows us the average test accuracy among all benign users but also provides the fairness of model performance through the distributions of test accuracy among all benign users.

As can be seen from Fig. 5, for the IMU dataset, FedCHAR outperforms other baselines by at least about 4% and at most by about 16% in the benign scenario. In attack scenarios, FedCHAR outperforms other baselines by at least about 7%, 4%, 6%, and 15%, and at most about 67%, 15%, 60%, and 60% over the four attack types, respectively. For the UWB dataset, in attack scenarios, FedCHAR outperforms other baselines by at least about 33%, 4%, 1%, and 19%, and at most about 81%, 26%, 68%, and 55% over the four attack types, respectively. In particular, the performance of FedCHAR is slightly lower than that of Ditto in the benign scenario. The reason is that the UWB data only contains eight users, and the similarity of the data distributions among all users is high, which cannot reflect the advantages of clustering. In practice, FedCHAR is reduced to Ditto when the number of clusters is 1. Therefore, in a practical scenario, we can choose the appropriate clustering threshold value to determine the number of clusters depending on the degree of heterogeneity among the user's data. For the FMCW dataset, in attack scenarios, the performance of FedCHAR in A1, A2, and A4 outperforms other baselines by at least about 3%, 3%, and 55%, and at most about 84%, 88%, and 89%. In particular, in the benign scenario, FedAvg performs best due to users' high similarity of data distributions. For the Depth dataset, FedCHAR outperforms other baselines

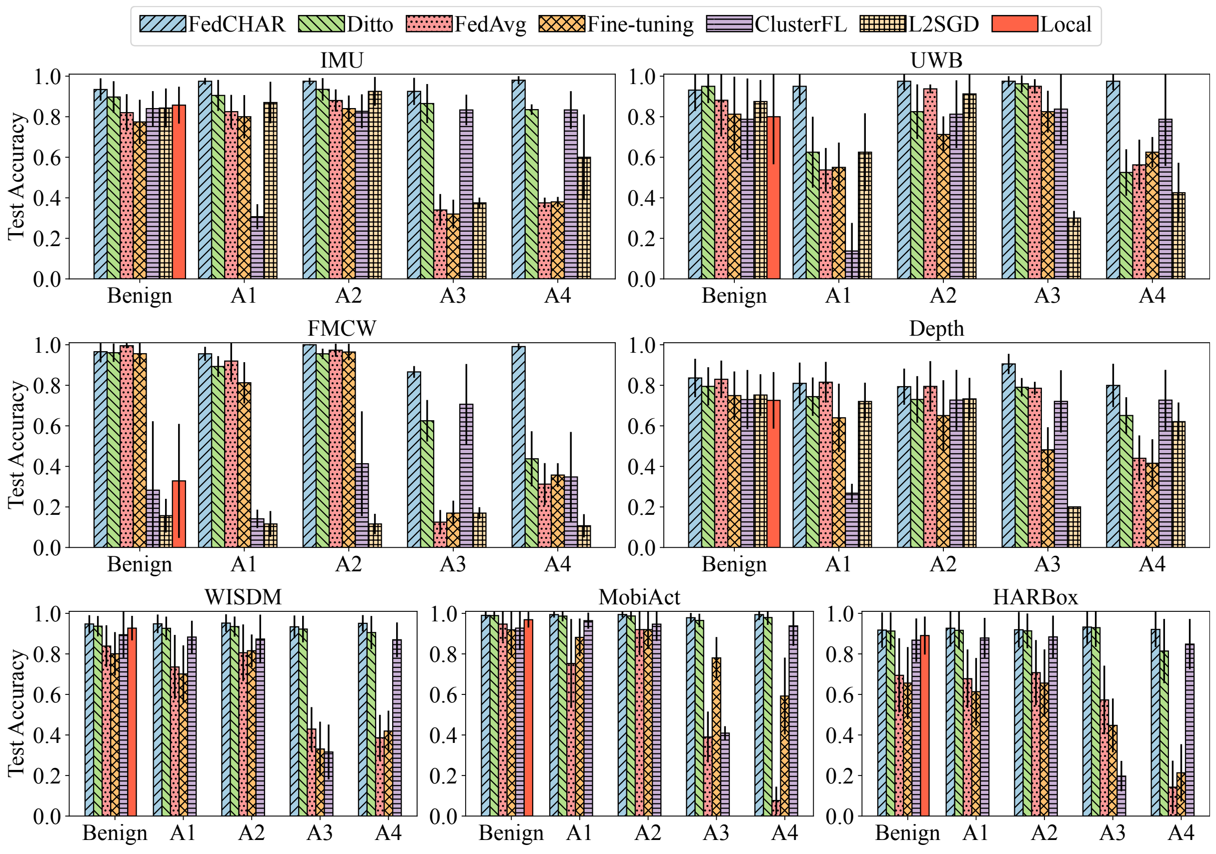


Fig. 5. Comparison of the performance of different baselines in benign and attack scenarios on seven datasets. The bar plot with error bars is used to show the average test accuracy of the models for all benign users and the fairness of the model performance among all benign users.

by at least 1% and at most by about 10% in the benign scenario. In attack scenarios, the performance of FedCHAR in A3 and A4 outperforms other baselines by at least about 11% and 7%, and at most about 70% and 38%. It is worth noting that the reason why the performance of most baselines is not ideal is that the user test set of the Depth dataset contains data from other users.

For the WISDM dataset, FedCHAR outperforms other baselines by at least about 1% and at most by about 15% in the benign scenario. In attack scenarios, FedCHAR outperforms other baselines by at least about 2%, 2%, 1%, and 5%, and at most about 25%, 15%, 62%, and 56% over the four attack types, respectively. For the MobiAct dataset, FedCHAR outperforms other baselines by at least about 0.1% and at most by about 7% in the benign scenario. In attack scenarios, FedCHAR outperforms other baselines by at least about 1%, 1%, 1%, and 2%, and at most about 24%, 7%, 59%, and 98% over the four attack types, respectively. For the HARBox dataset, FedCHAR outperforms other baselines by at least about 0.4% and at most by about 26% in the benign scenario. In attack scenarios, FedCHAR outperforms other baselines by at least about 1%, 1%, 0.3%, and 11%, and at most about 31%, 27%, 73%, and 78% over the four attack types, respectively.

To sum up, the above results show that FedCHAR can maintain relatively high accuracy and low variance in the benign scenario. A plausible explanation is that FedCHAR clusters users with similar data distributions, thereby reducing the impact of user data heterogeneity while improving the accuracy and fairness of model performance among users. In attack scenarios, FedCHAR can defend against all types of attacks, while other baselines have limited generalization ability to defend against various attacks. The reason is that FedCHAR can isolate benign nodes from malicious nodes to a certain extent, significantly reducing the number of malicious nodes in the clusters where benign nodes are located. So the attack will not dominate the performance of the user model. Even if there are malicious nodes that FedCHAR does not isolate, they will not negatively impact the training process because of their high similarity to benign users (i.e., similar updating direction of their model parameters). Moreover, combining robust aggregation methods can further reduce the impact of attacks. We will discuss this in-depth in Section 7.5.

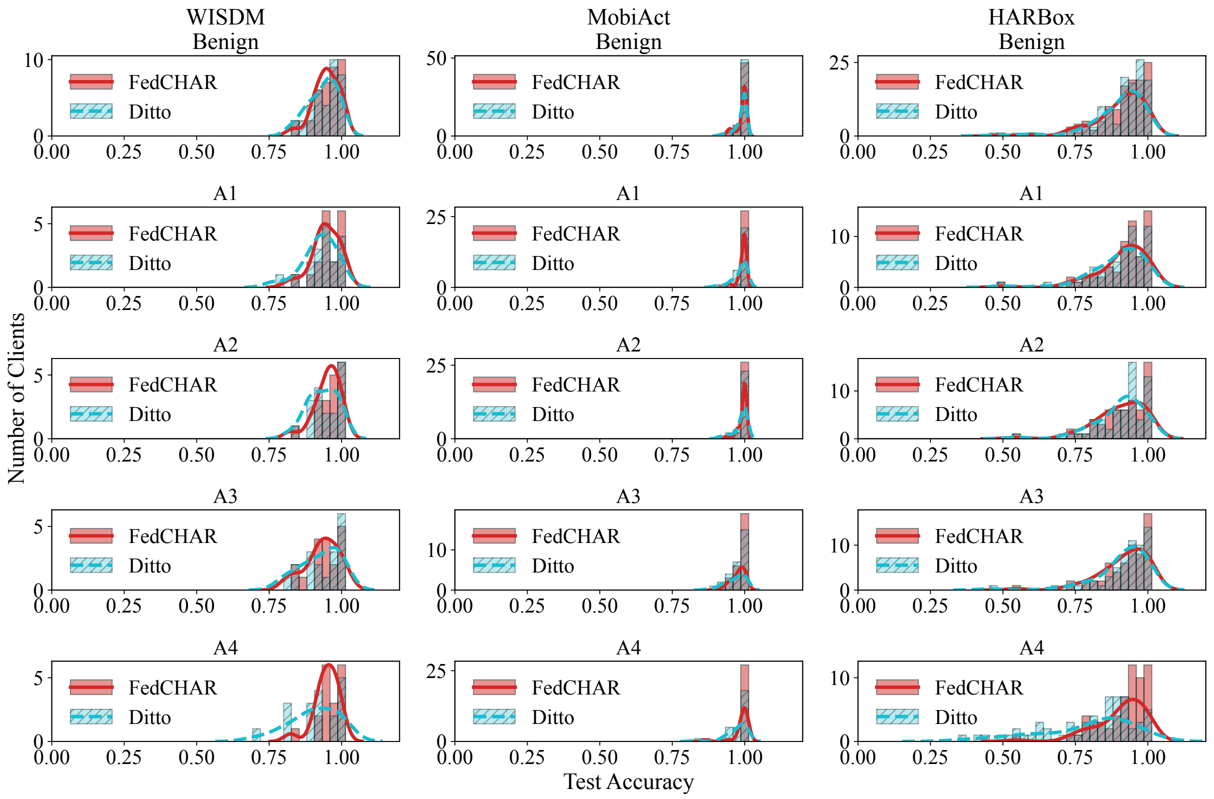


Fig. 6. Comparison of the distribution of test accuracy for all benign user models in FedCHAR and Ditto.

7.2 Fairness

In order to further demonstrate that FedCHAR is beneficial to improve the fairness of the model performance among users, we select three large-scale datasets (i.e., WISDM, MobiAct, and HARBox) and compare FedCHAR with Ditto, which is the closest to our work. Specifically, we use three datasets to show the distribution of test accuracy for all benign user models in benign scenario and attack scenarios. In Fig. 6, the height of the bar

Table 3. Complete statistics on the distribution of test accuracy in FedCHAR and Ditto.

Dataset	Method	Type	Benign	A1(50%)	A2(50%)	A3(50%)	A4(50%)
WISDM	FedCHAR	Average(variance)	**0.948(1.88e-3)**	**0.948(2.11e-3)**	**0.952(1.89e-3)**	**0.934(3.13e-3)**	**0.951(1.65e-3)**
		Worst 10%	**0.852**	**0.824**	0.824	**0.821**	**0.824**
		Best 10%	1.0	1.0	1.0	1.0	1.0
	Ditto	Average(variance)	0.937(2.53e-3)	0.926(3.34e-3)	0.935(2.46e-3)	0.923(4.30e-3)	0.905(6.67e-3)
		Worst 10%	0.833	0.765	0.824	0.793	0.706
		Best 10%	1.0	1.0	1.0	1.0	1.0
MobiAct	FedCHAR	Average(variance)	**0.990(3.34e-4)**	**0.994(2.09e-4)**	**0.994(2.03e-4)**	**0.978(5.57e-4)**	**0.994(7.08e-4)**
		Worst 10%	**0.945**	**0.946**	**0.953**	**0.926**	**0.903**
		Best 10%	1.0	1.0	1.0	1.0	1.0
	Ditto	Average(variance)	0.989(3.72e-4)	0.987(6.69e-4)	0.987(5.53e-4)	0.965(1.10e-3)	0.979(1.29e-3)
		Worst 10%	0.938	0.912	0.924	0.894	0.878
		Best 10%	1.0	1.0	1.0	1.0	1.0
HARBox	FedCHAR	Average(variance)	**0.917(7.59e-3)**	**0.927(8.08e-3)**	**0.919(7.68e-3)**	**0.932(7.52e-3)**	**0.921(7.58e-3)**
		Worst 10%	**0.715**	**0.711**	0.720	**0.727**	**0.721**
		Best 10%	1.0	1.0	1.0	1.0	**1.0**
	Ditto	Average(variance)	0.913(8.54e-3)	0.916(8.66e-3)	0.914(7.34e-3)	0.929(8.98e-3)	0.814(2.51e-2)
		Worst 10%	0.694	0.698	0.720	0.694	0.458
		Best 10%	1.0	1.0	1.0	1.0	0.991

represents the number of users. The curve reflects the distribution of test accuracy for all benign user models. The steeper the curve and the farther to the right, the more uniform and higher the performance among user models. As seen from Fig. 6, in almost all cases, the performance of user models in FedCHAR is more uniform and higher than that of Ditto. In particular, in the scenario where the attack type is A4, the performance distribution for all benign users in Ditto is not uniform, and the performance of a large number of user models is significantly reduced due to the impact of the attack. FedCHAR maintains the fairness of the model performance among all benign users while ensuring that the performance of the vast majority of user models is protected from attacks.

Additionally, we provide complete statistics on the distribution of test accuracy in FedCHAR and Ditto. We report the average test accuracy, the worst 10%, and the best 10% of all the user models for both FedCHAR and Ditto using the three large-scale datasets in the benign and attack scenarios. Among them, the average test accuracy of the 10% user models with the lowest performance and the variance of the test accuracy among the user models can further reflect the fairness. Only the average test accuracy of user models does not fully reflect the performance distribution of user models because there may be a few user models with particularly low test accuracies compared to others. In the FL-based HAR scenario, our aim is that all user models have satisfactory performance and maintain a certain uniformity. Specifically, as shown in Table 3, the average test accuracy, the worst 10%, and the best 10% in FedCHAR are better than Ditto in all cases. FedCHAR can guarantee both high accuracy and fairness of user models in most cases. In particular, in the scenario where the attack type is A4, the average test accuracy of the 10% user models with the lowest performance in Ditto is only 45.8%, indicating that the performance of some user models is greatly affected by the attack.

To sum up, according to the above analysis, FedCHAR can simultaneously improve the robustness, fairness, and accuracy of the user models compared to Ditto in both benign and attack scenarios. A reasonable explanation is that Ditto only improves the fairness of user model performance through personalization. When the training process of the global model is invaded by the attack, the training of the user-personalized model will be affected by the attacked global model, resulting in model performance degradation. Although Ditto reduces the reference degree of the user-personalized model to the global model by reducing the value of λ, the small amount of local user data is not enough to train a model with satisfactory performance. Therefore, in attack scenarios, weighing the robustness and accuracy of the user model is a difficult problem faced by Ditto. However, FedCHAR can not only assign users with similar data distribution to the same cluster through clustering but also isolate benign nodes from a part of malicious nodes simultaneously. As the number of malicious nodes decreases, the impact of the attack on the global model is weakened. In addition, FedCHAR further reduces the interference of the attacked global model by training the user's local model in a personalized manner within the cluster.

7.3 Ability to Isolate Benign Nodes from Malicious Nodes

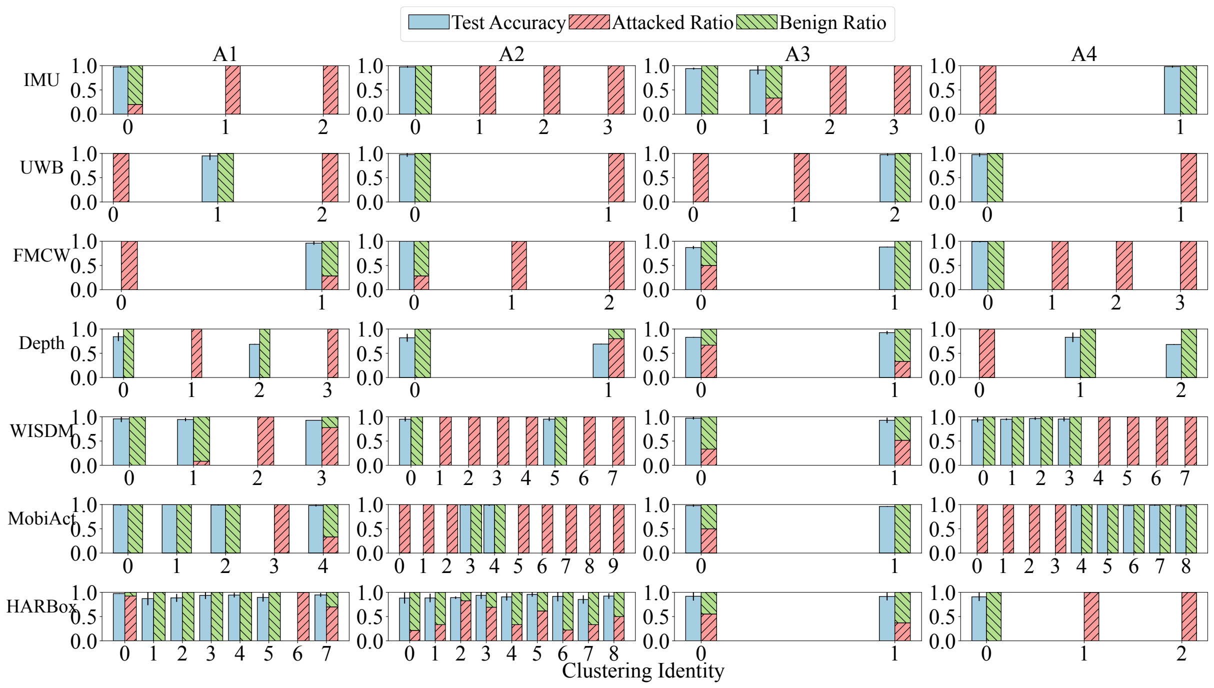


Fig. 7. The accuracy, fairness, and ability to isolate the benign nodes from malicious nodes (i.e., robustness) of FedCHAR under different types of attacks are evaluated on seven datasets of different sizes. In most cases FedCHAR isolates the benign nodes from some or even all of the malicious nodes, greatly reducing the number of malicious nodes in the clusters where benign nodes are located and mitigating the impact of the attack.

In this section, to more clearly show the extent to which FedCHAR isolates benign nodes from malicious nodes through clustering, we show three elements simultaneously through bar charts: the average test accuracy of benign user models within each cluster and distribution and the ratio of the number of malicious nodes to the number of benign nodes within each cluster.

As seen from Fig. 7, FedCHAR can isolate benign nodes from malicious nodes to varying degrees in each dataset. Specifically, for the IMU, UWB, FMCW, and MobiAct datasets, FedCHAR isolates benign nodes from the majority of malicious nodes, significantly reducing the number of malicious nodes in the clusters where benign nodes are located and thus directly mitigating the impact of attacks on the user models. In particular, in the Depth, WISDM, and HARBox datasets, the number of malicious nodes within some clusters is much higher than that of benign nodes. However, the average test accuracy of benign nodes within clusters still maintains high values. For example, in the HARBox dataset, for attack type A1, malicious nodes are predominant in clusters 0 and 7, but the performance of the benign user models remains unaffected by the attacks. A plausible explanation is that FedCHAR performs clustering based on calculating the similarity of the user-uploaded model update parameters, which reflect the updating direction of the model. Based on this, as they are similar to the tampered model update parameters uploaded by the malicious nodes, some benign nodes are assigned to the cluster where the malicious nodes are located, representing that their model updating directions are similar. Therefore, the model performance of the benign node is not affected by the attack.

Specifically, in the scenario where the attack type is A4, FedCHAR isolates benign nodes from all the malicious nodes on all datasets, completely resisting the attack. In contrast, as demonstrated in the previous sections, other methods are generally less resilient to A4. The reason is that the global model updating direction is severely disturbed by the malicious nodes uploading opposite model update parameters. FedCHAR can accurately distinguish the opposite model update parameters uploaded by malicious nodes by calculating the similarity of model update parameters among users. In addition, although FedCHAR fails to isolate benign nodes from some of the malicious nodes through clustering (e.g., in the case of the HARBox dataset with attack type A2), FedCHAR still attenuates the interference of the global model by personalizing the training of the user's local model within the cluster. We can also choose a suitable robust aggregation method to further defend against the attack.

In summary, FedCHAR can isolate benign nodes from a large number of malicious nodes through clustering, significantly reducing the number of malicious nodes in the clusters where benign nodes are located. By calculating the similarity of the model update parameters among users, FedCHAR can subtly prevent the model updating direction of benign nodes from being changed by the attack to some extent.

7.4 Comparison of Different Robust Aggregation Methods

In order to further examine the robustness of FedCHAR, we use five datasets (i.e., IMU, UWB, WISDM, MobiAct, and HARBox) to compare the performance of FedCHAR with various commonly used robust aggregation methods under four attack scenarios with malicious node proportions of 20% and 50%, respectively. The robust baselines consist of FedAvg combined with the robust aggregation methods. In addition, we use FedAvg, which is not combined with any robust aggregation method, as a reference to explore the impact of robust aggregation methods.

As can be seen from Table 4, in the benign scenario, the model performance of most robust aggregation methods is degraded compared to the original FedAvg. In attack scenarios with a 20% attack ratio, for attack types A1 and A2, K-Norm and Multi-Krum, in most cases, improve the model performance compared to the original FedAvg. For attack type A3, Multi-Krum, Clipping, and K-Norm, in most cases, improve model performance over raw FedAvg.

In the scenario where the proportion of attacks is 50%, Multi-Krum, K-Norm, and Median show more robust against attack type A1. For attack type A2, most robust aggregation methods show a decrease in model performance compared to the original FedAvg. For attack type A3, the model performance of the original FedAvg is significantly degraded by the attack. Krum, Multi-Krum, and K-Norm are more resilient to the attack. For attack type A4, Multi-Krum and K-Norm show strong robustness in some datasets. Specifically, in the UWB dataset, the Multi-Krum improves the model performance by 25% compared to the original FedAvg. In the MobiAct dataset, K-Norm achieves a 40% performance improvement over the original FedAvg model. However, in the rest of the datasets, most of the robust aggregation methods are still affected by the attack, despite the performance of the model is improved compared to the original FedAvg.

In summary, in the benign scenario, most robust aggregation methods have degraded model performance compared to the original FedAvg. A plausible explanation is that the robust aggregation methods select some user models instead of all user models for aggregation, which may filter out some valuable updates. In attack scenarios, Multi-Krum, K-Norm, and Median are able to resist the impact of the attack to some extent in most cases. In

Table 4. Comparison of the performance of different robust aggregation methods on five datasets in benign and attack scenarios.

Methods	**Datasets**	**Benign**	**A1**		**A2**		**A3**		**A4**	
			20%	50%	20%	50%	20%	50%	20%	50%
FedAvg	IMU	.820(.0083)	.810(.0104)	.825(.0069)	.837(.0089)	.880(.0030)	.760(.0247)	.340(.0062)	.810(.0226)	.375(.0007)
	UWB	.881(.0318)	**.950(.0071)**	.537(.0117)	.943(.0196)	.938(.0005)	.950(.0021)	.950(.0012)	.900(.0236)	.562(.0155)
	WISDM	.839(.0106)	.797(.0218)	.736(.0250)	.824(.0099)	.806(.0195)	.662(.0236)	.429(.0117)	.819(.0142)	.386(.0128)
	MobiAct	.947(.0071)	.917(.0165)	.753(.0481)	.952(.0024)	.920(.0160)	.815(.0266)	.390(.0157)	.922(.0091)	.077(.0047)
	HARBox	.694(.0335)	.722(.0252)	.678(.0208)	.713(.0244)	.707(.0259)	.687(.0286)	.573(.0286)	.658(.0306)	.143(.0170)
Krum	IMU	.554(.1010)	.513(.0730)	.510(.0849)	.473(.0313)	.510(.0905)	.583(.0246)	.665(.0383)	.460(.0340)	.480(.1080)
	UWB	.844(.0403)	.900(.0279)	.537(.0105)	.886(.0277)	.475(.0106)	.914(.0298)	.912(.0030)	.750(.0207)	.425(.0019)
	WISDM	.727(.0249)	.786(.0246)	.375(.0367)	.727(.0303)	.712(.0342)	.703(.0287)	.817(.0037)	.705(.0316)	.358(.0157)
	MobiAct	.613(.0370)	.669(.0498)	.074(.0156)	.780(.0588)	.818(.0385)	.682(.0401)	.801(.0493)	.781(.0368)	.047(.0001)
	HARBox	.389(.0368)	.411(.0411)	.431(.0354)	.402(.0464)	.378(.0449)	.405(.0363)	.416(.0383)	.415(.0439)	.360(.0481)
Multi-Krum	IMU	.820(.0083)	.780(.0100)	.650(.0509)	.830(.0078)	.645(.0477)	.837(.0093)	.900(.0038)	.757(.0295)	.575(.0907)
	UWB	.881(.0318)	.936(.0191)	.725(.0056)	.929(.0192)	.975(.0019)	.936(.0191)	**1.000(0.0)**	.921(.0199)	.812(.0042)
	WISDM	.825(.0109)	.856(.0055)	.727(.0258)	.825(.0117)	.820(.0140)	.816(.0161)	.829(.0037)	.844(.0088)	.739(.0374)
	MobiAct	.940(.0110)	.932(.0134)	.796(.0169)	.931(.0147)	.855(.0439)	.946(.0089)	.947(.0063)	.905(.0270)	.646(.0532)
	HARBox	.695(.0339)	.722(.0246)	.749(.0272)	.714(.0290)	.728(.0268)	.737(.0257)	.756(.0281)	.693(.0266)	.567(.0371)
Clipping	IMU	.794(.0083)	.793(.0108)	.810(.0081)	.833(.0073)	.855(.0045)	.813(.0080)	.820(.0090)	.807(.0142)	.400(.0018)
	UWB	.881(.0375)	.929(.0192)	.600(.0013)	.936(.0191)	.812(.0092)	.936(.0191)	.975(.0006)	.893(.0239)	.475(.0106)
	WISDM	.842(.0067)	.820(.0249)	.780(.0461)	.838(.0091)	.823(.0168)	.835(.0146)	.759(.0069)	.831(.0084)	.504(.0325)
	MobiAct	.958(.0023)	.946(.0035)	.788(.0287)	.932(.0111)	.943(.0085)	.939(.0069)	.903(.0137)	.920(.0143)	.524(.0440)
	HARBox	.682(.0361)	.716(.0275)	.681(.0205)	.715(.0251)	.703(.0307)	.714(.0290)	.757(.0237)	.679(.0271)	.239(.0260)
K-Norm	IMU	.820(.0083)	.820(.0100)	.630(.0243)	.847(.0126)	.645(.0477)	.837(.0093)	.900(.0038)	.757(.0295)	.605(.0885)
	UWB	.881(.0318)	.943(.0196)	.762(.0042)	.929(.0192)	.362(.0017)	.936(.0191)	**1.000(0.0)**	.900(.0179)	.300(.0013)
	WISDM	.827(.0122)	.851(.0083)	.807(.0135)	.826(.0160)	.828(.0188)	.823(.0095)	.800(.0131)	.832(.0176)	.793(.0048)
	MobiAct	.944(.0097)	.930(.0087)	.821(.0195)	.918(.0272)	.858(.0486)	.932(.0121)	.927(.0106)	.892(.0265)	.497(.0348)
	HARBox	.692(.0336)	.725(.0273)	.749(.0271)	.716(.0294)	.722(.0281)	.739(.0238)	.757(.0278)	.687(.0264)	.551(.0395)
Median	IMU	.763(.0095)	.800(.0139)	.835(.0133)	.817(.0143)	.845(.0183)	.800(.0075)	.845(.0161)	.777(.0245)	.650(.0059)
	UWB	.887(.0286)	.929(.0185)	.688(.0017)	.921(.0199)	.738(.0080)	.943(.0146)	.988(.0005)	.893(.0289)	.562(.0067)
	WISDM	.801(.0247)	.791(.0200)	.726(.0269)	.819(.0175)	.814(.0163)	.828(.0166)	.781(.0113)	.817(.0171)	.328(.0185)
	MobiAct	.907(.0282)	.858(.0274)	.837(.0319)	.907(.0209)	.917(.0270)	.897(.0253)	.875(.0423)	.887(.0327)	.563(.0460)
	HARBox	.649(.0385)	.687(.0316)	.677(.0308)	.680(.0277)	.661(.0264)	.689(.0328)	.737(.0291)	.648(.0274)	.251(.0112)
FedCHAR	IMU	**.934(.0031)**	**.933(.0016)**	**.975(.0003)**	**.937(.0027)**	**.975(.0003)**	**.903(.0062)**	**.925(.0045)**	**.937(.0014)**	**.980(.0004)**
	UWB	**.931(.0106)**	.950(.0079)	**.950(.0075)**	**.964(.0034)**	**.975(.0019)**	**.986(.0012)**	.975(.0006)	**.964(.0034)**	**.975(.0019)**
	WISDM	**.948(.0019)**	**.947(.0019)**	**.948(.0021)**	**.941(.0030)**	**.952(.0019)**	**.903(.0068)**	**.900(.0024)**	**.941(.0030)**	**.951(.0017)**
	MobiAct	**.990(.0033)**	**.989(.0004)**	**.994(.0002)**	**.987(.0005)**	**.994(.0002)**	**.979(.0009)**	**.978(.0006)**	**.988(.0005)**	**.994(.0007)**
	HARBox	**.917(.0076)**	**.911(.0075)**	**.927(.0081)**	**.909(.0081)**	**.919(.0077)**	**.910(.0082)**	**.932(.0075)**	**.908(.0078)**	**.921(.0076)**

particular, when the attack type is A3, Multi-Krum and K-Norm have significantly better model performance than the original FedAvg and are robust against the attack.

7.5 FedCHAR Augmented with Robust Aggregation Methods

In this work, FedCHAR uses the average aggregation method by default to aggregate user models. In this section, to explore the impact of aggregation methods on FedCHAR, we combine FedCHAR with several robust aggregation methods. We use five datasets (i.e., UWB, IMU, WISDM, MobiAct, and HARBox) to compare the model performance of FedCHAR under the benign scenario and attack scenarios with 20% and 50% of malicious nodes.

As seen from Table 5, the performance of FedCHAR can approach or even exceed that of most FedCHAR combined with robust aggregation methods. Specifically, for the IMU dataset, the performance of FedCHAR combined with Krum and Clipping is improved compared to the original FedCHAR when the attack type is A3 and the ratios of malicious nodes are 20% and 50%. A plausible explanation is that A3 attacks cause more interference in model training, and Krum reduces the interference of the poisoned model updates uploaded by

Table 5. Comparison of the performance of FedCHAR combined with robust aggregation methods on five datasets in benign and attack scenarios.

Methods	Datasets	Benign	A1		A2		A3		A4	
			20%	50%	20%	50%	20%	50%	20%	50%
FedCHAR	IMU	**.934(.0031)**	**.933(.0016)**	**.975(.0003)**	**.937(.0027)**	**.975(.0003)**	.903(.0062)	.925(.0045)	**.937(.0014)**	**.980(.0004)**
	UWB	.931(.0106)	.950(.0079)	.950(.0075)	**.964(.0034)**	**.975(.0019)**	**.986(.0012)**	.975(.0006)	**.964(.0034)**	.975(.0019)
	WISDM	**.948(.0019)**	.947(.0019)	**.948(.0021)**	.941(.0030)	**.952(.0019)**	.903(.0068)	**.900(.0024)**	**.941(.0030)**	**.951(.0017)**
	MobiAct	**.990(.0033)**	.989(.0004)	**.994(.0002)**	**.987(.0005)**	**.994(.0002)**	**.979(.0009)**	.978(.0006)	**.988(.0005)**	**.994(.0007)**
	HARBox	**.917(.0076)**	**.911(.0075)**	**.927(.0081)**	**.909(.0081)**	**.919(.0077)**	**.910(.0082)**	**.932(.0075)**	**.908(.0078)**	**.921(.0076)**
FedCHAR + Krum	IMU	.769(.0200)	.730(.0329)	.835(.0103)	.910(.0056)	.960(.0008)	.887(.0116)	**.965(.0015)**	.877(.0090)	.955(.0013)
	UWB	.850(.0300)	**.950(.0071)**	.825(.0756)	.936(.0077)	.863(.0080)	.943(.0046)	.838(.0155)	.950(.0079)	**.988(.0005)**
	WISDM	.925(.0020)	.926(.0033)	.919(.0026)	.915(.0064)	.894(.0123)	.851(.0154)	.670(.0323)	.928(.0036)	.913(.0045)
	MobiAct	.972(.0018)	.963(.0051)	.953(.0028)	.959(.0024)	.967(.0022)	.963(.0038)	.919(.0312)	.959(.0067)	.961(.0064)
	HARBox	.883(.0106)	.885(.0091)	.905(.0079)	.876(.0090)	.893(.0100)	.884(.0087)	.895(.0095)	.876(.0085)	.888(.0111)
FedCHAR + Multi-Krum	IMU	.934(.0031)	.910(.0034)	.885(.0051)	.867(.0096)	.970(.0005)	.910(.0037)	.875(.0231)	.840(.0119)	.975(.0007)
	UWB	.931(.0106)	.950(.0079)	**.963(.0042)**	.964(.0034)	.938(.0030)	.957(.0046)	.975(.0019)	.964(.0034)	**.988(.0005)**
	WISDM	.938(.0028)	.941(.0021)	.926(.0023)	.917(.0137)	.938(.0018)	.926(.0029)	.893(.0048)	.939(.0030)	.937(.0025)
	MobiAct	.987(.0004)	**.990(.0010)**	.987(.0005)	.983(.0008)	.987(.0006)	.963(.0047)	**.968(.0051)**	.977(.0025)	.976(.0014)
	HARBox	.913(.0083)	.898(.0086)	.919(.0082)	.905(.0085)	.918(.0067)	.909(.0086)	.925(.0088)	.904(.0095)	.913(.0077)
FedCHAR + Clipping	IMU	.906(.0067)	.930(.0020)	.945(.0011)	.867(.0100)	.950(.0029)	**.920(.0037)**	.935(.0027)	.867(.0081)	.965(.0005)
	UWB	.919(.0014)	.836(.0455)	.938(.0117)	.943(.0089)	.963(.0042)	.921(.0199)	**.988(.0005)**	.943(.0089)	.950(.0075)
	WISDM	.931(.0023)	**.948(.0018)**	.892(.0249)	**.947(.0021)**	.930(.0030)	**.940(.0019)**	.846(.0141)	.932(.0035)	.934(.0028)
	MobiAct	.990(.0004)	.988(.0005)	.986(.0008)	.984(.0006)	.987(.0004)	.979(.0011)	.954(.0049)	.975(.0048)	.981(.0047)
	HARBox	.913(.0081)	.903(.0083)	.915(.0084)	.902(.0078)	.914(.0084)	.904(.0079)	.930(.0087)	.902(.0079)	.920(.0084)
FedCHAR + K-Norm	IMU	.934(.0031)	.910(.0034)	.860(.0074)	.867(.0096)	.905(.0065)	.910(.0037)	.905(.0069)	.840(.0119)	.905(.0043)
	UWB	.931(.0106)	.836(.0462)	.812(.0142)	.886(.0162)	.812(.0205)	.957(.0046)	.975(.0019)	.886(.0162)	.787(.0230)
	WISDM	.940(.0019)	.938(.0029)	.923(.0037)	.932(.0027)	.947(.0014)	.874(.0015)	.887(.0042)	.939(.0025)	.931(.0031)
	MobiAct	.985(.0006)	.979(.0038)	.985(.0008)	.984(.0007)	.984(.0007)	.968(.0033)	.958(.0132)	.975(.0032)	.983(.0009)
	HARBox	.912(.0082)	.902(.0083)	.918(.0076)	.904(.0076)	.920(.0078)	.907(.0090)	.924(.0081)	.871(.0252)	.911(.0089)
FedCHAR + Median	IMU	.906(.0052)	.930(.0024)	.960(.0008)	.897(.0057)	.935(.0063)	.920(.0041)	.940(.0026)	.867(.0082)	.945(.0039)
	UWB	**.944(.0059)**	.943(.0075)	.650(.0338)	.964(.0034)	.975(.0019)	.921(.0199)	.925(.0056)	.964(.0034)	.975(.0019)
	WISDM	.919(.0029)	.938(.0019)	.918(.0097)	.943(.0022)	.932(.0019)	.922(.0033)	.873(.0066)	.937(.0022)	.938(.0023)
	MobiAct	.973(.0044)	.982(.0008)	.988(.0006)	.968(.0040)	.984(.0011)	.974(.0016)	.941(.0125)	.973(.0038)	.974(.0036)
	HARBox	.907(.0081)	.901(.0092)	.915(.0072)	.900(.0094)	.909(.0105)	.904(.0085)	.916(.0082)	.903(.0092)	.915(.0076)

malicious nodes through selecting the user model update with the highest similarity to other model updates as global model updates. Clipping controls the size of the model update parameters to defend against gradient amplification attacks (i.e., A3). For the UWB dataset, the performance of FedCHAR combined with Multi-Krum is improved compared to the original FedCHAR when the attack type is A1 and A4 with the attack percentage is 50%. The reason is that malicious nodes upload opposite update parameters, and Multi-Krum can suppress model updates in the opposite direction by taking the average of the first k minimum scores. For the WISDM dataset, when the attack type is A3, the performance of FedCHAR combined with Clipping is improved by about 4%, further verifying that the Clipping method is robust against A3 attacks. For the MobiAct dataset, the performance of FedCHAR combined with Multi-Krum is improved compared to the original FedCHAR when the attack type is A1 at 20%, and the attack type is A3 at 50%, further validating the robustness of the Multi-Krum method against attacks. For the HARBox dataset, the performance of the original FedCHAR outperforms all the combined robust aggregation methods. Furthermore, FedCHAR combined with the K-Norm method does not appear to result in a performance gain, as the K-norm method reduces the interference of the poisoned model updates in model updates aggregation by averaging the k L2-Norm minimum user model updates, potentially losing some valuable updating information.

In summary, the original FedCHAR outperforms the combined robust aggregation method in most cases, because FedCHAR can defend against attacks through both clustering and personalization (i.e., control the value

Table 6. Comparison of the performance of FedCHAR on three large-scale datasets with the other four baselines in the hybrid attack scenario.

Method	**Hybrid Attack (50%)**		
	WISDM	MobiAct	HARBox
FedCHAR	**0.935(2.27e-3)**	**0.990(3.12e-4)**	**0.930(8.08e-3)**
Ditto	0.908(2.27e-3)	0.974(1.33e-3)	0.929(7.94e-3)
FedAvg	0.770(3.97e-3)	0.896(1.49e-2)	0.729(2.68e-2)
Fine-tuning	0.692(1.73e-2)	0.783(3.52e-2)	0.708(3.07e-2)
L2SGD	0.851(2.77e-2)	0.926(1.32e-2)	0.887(9.18e-3)

Table 7. Comparison of performance of FedCHAR on seven datasets using three linkage criteria in benign and attack scenarios.

Dataset	**Linkage**	**Benign**	**A1(50%)**	**A2(50%)**	**A3(50%)**	**A4(50%)**
	Average	**0.934(3.05e-3)**	0.950(2.10e-3)	0.960(6.00e-4)	**0.925(4.48e-3)**	**0.980(4.00e-4)**
IMU	Complete	**0.934(3.05e-3)**	0.950(2.10e-3)	0.960(6.00e-4)	**0.925(4.48e-3)**	**0.980(4.00e-4)**
	Single	**0.934(3.05e-3)**	**0.975(2.75e-4)**	**0.975(2.75e-4)**	0.920(8.80e-3)	**0.980(4.00e-4)**
	Average	0.925(7.50e-3)	**0.950(7.50e-3)**	**0.975(1.87e-3)**	0.975(6.25e-4)	**0.975(1.87e-3)**
UWB	Complete	0.925(7.50e-3)	**0.950(7.50e-3)**	**0.975(1.87e-3)**	0.975(6.25e-4)	**0.975(1.87e-3)**
	Single	**0.931(1.06e-2)**	**0.950(7.50e-3)**	**0.975(1.87e-3)**	**0.975(5.25e-4)**	**0.975(1.87e-3)**
	Average	0.956(2.25e-3)	0.920(3.26e-5)	**1.000(0.0)**	0.446(1.44e-1)	**0.991(2.19e-4)**
FMCW	Complete	**0.966(2.76e-3)**	0.929(2.05e-3)	0.991(2.19e-4)	0.491(1.34e-1)	0.982(3.57e-4)
	Single	0.956(2.69e-3)	**0.955(1.20e-3)**	0.991(2.19e-4)	**0.688(2.59e-2)**	**0.991(2.19e-4)**
	Average	**0.836(9.04e-3)**	0.799(1.05e-2)	0.792(1.15e-2)	**0.905(2.49e-3)**	**0.799(1.14e-2)**
Depth	Complete	**0.836(8.97e-3)**	0.800(1.09e-2)	**0.793(7.95e-3)**	0.904(2.58e-3)	0.796(1.12e-2)
	Single	**0.836(8.84e-3)**	**0.801(1.05e-2)**	0.789(1.22e-2)	0.901(4.81e-3)	0.798(1.18e-2)
	Average	0.933(2.24e-3)	0.937(2.90e-3)	0.931(3.64e-3)	0.871(7,24e-3)	0.927(2.42e-3)
WISDM	Complete	**0.948(1.88e-3)**	**0.948(2.11e-3)**	**0.952(1.89e-3)**	**0.900(2.37e-3)**	**0.951(1.65e-3)**
	Single	0.925(4.11e-3)	0.926(3.16e-3)	0.923(2.95e-3)	0.879(5.46e-3)	0.916(3.30e-3)
	Average	0.985(8.38e-4)	0.987(5.71e-4)	0.992(4.10e-4)	0.938(9.67e-4)	0.988(5.61e-4)
MobiAct	Complete	**0.990(3.34e-4)**	**0.994(2.09e-4)**	**0.994(2.03e-4)**	**0.978(5.57e-4)**	**0.994(7.08e-4)**
	Single	0.988(4.20e-4)	0.988(6.77e-4)	0.988(4.19e-4)	0.929(2.06e-3)	0.985(6.91e-4)
	Average	0.910(7.62e-3)	0.917(7.46e-3)	0.917(7.67e-3)	0.932(7.86e-3)	0.920(7.62e-3)
HARBox	Complete	**0.917(7.59e-3)**	**0.927(8.08e-3)**	**0.919(7.68e-3)**	**0.932(7.52e-3)**	**0.921(7.58e-3)**
	Single	0.908(8.43e-3)	0.911(8.32e-3)	0.909(8.37e-3)	0.925(8.05e-3)	0.921(8.07e-3)

of the regular coefficient λ). For certain types of attacks, FedCHAR combined with Krum, Multi-Krum, and Clipping is more likely to enhance the robustness of the system and bring performance improvements.

7.6 Hybrid Attack

To further mimic realistic attack scenarios, we explore the robustness, fairness, and accuracy of FedCHAR in the hybrid attack scenario. We assume that the ratio of malicious nodes is 50%, and then we arbitrarily assign the malicious nodes one of the four types of attacks to be launched. As seen from Table 6, FedCHAR still outperforms other baselines in the hybrid attack scenario, further validating the ability of FedCHAR to simultaneously improve the robustness, fairness, and accuracy of the FL system.

7.7 Impact of Linkage Criteria

To explore the influence of linkage criteria in agglomerative hierarchical clustering on the final performance, we report the performance of FedCHAR on seven datasets using three linkage criteria in benign and attack scenarios.

As seen from Table 7, for the small-scale datasets (i.e., IMU, UWB, FMCW, and Depth), the difference in performance among the three linkage criteria is not obvious. For the UWB dataset, using the single linkage as a measure of inter-cluster distance performs better than other linkages. For the IMU dataset, using single linkage as a measure of inter-cluster distance performs better than other linkages, except for attack type A3. When the attack type is A3, it seems more appropriate to use the other two linkages. The reason is that the single linkage is prone to chain reactions. For example, when the model update parameters of benign user Alice are similar to the tampered model update parameters uploaded by the malicious nodes, Alice joins the cluster where the malicious nodes are located. At the subsequent clustering, benign user Bob, whose model update parameters are similar to those of Alice, is also assigned to the cluster where the malicious node is located, thus creating an unintended chain effect. For the FMCW dataset, using the single linkage performs better in most cases relative to the other two linkages. For the Depth dataset, using average linkage performs better in most cases compared to the other two linkages. For the large-scale datasets (i.e., WISDM, MobiAct, and HARBox), the performance of using complete linkage as a measure of inter-cluster distance is better than the other linkages.

In summary, due to the degree of bias in the results for small-scale datasets, we recommend using complete linkage as a measure of inter-cluster distance after validation on large-scale datasets such as WISDM, MobiAct, and HARBox. A plausible explanation is that the complete linkage takes the distance between the two data points with the farthest distance in different clusters as the inter-cluster distance, which makes the distance between the point to be clustered and all points in a cluster as small as possible.

7.8 Impact of Clustering Threshold σ

In this section, we explore the impact of different values of clustering threshold σ on the performance of FedCHAR. In agglomerative hierarchical clustering, the number of clusters can be determined automatically by the clustering threshold. When the similarity between clusters is greater than the clustering threshold σ, the two clusters are merged. Obviously, the larger the clustering threshold σ, the more difficult it is to merge the two clusters, and the more clusters are left in the end. Therefore, in this work, we regard the clustering threshold σ as a hyperparameter to perform a large number of searches and obtain the results.

By analyzing Fig. 8, we believe it is more appropriate to choose a smaller value of clustering threshold for small-scale datasets (i.e., IMU, UWB, FMCW, and Depth). For large-scale datasets (i.e., WISDM, MobiAct, and HARBox), a moderate value of the clustering threshold is more appropriate. Because too small a clustering threshold may result in less similar users being assigned to the same cluster. The global model performance within the cluster is easily degraded by user data heterogeneity, which in turn leads to a degradation in the performance of the user-personalized models. In attack scenarios, if the value of clustering threshold is too small, most benign nodes cannot be separated from malicious nodes, making the updating of the global model within the cluster vulnerable to attack. A moderate cluster threshold not only helps to isolate benign nodes from malicious nodes but also reduces the possibility of benign nodes being assigned to clusters with a large number of malicious nodes.

In summary, choosing a moderate clustering threshold is a good trade-off between the generalizability of features learned by the global model within a cluster and the accuracy of isolating the malicious nodes from benign nodes.

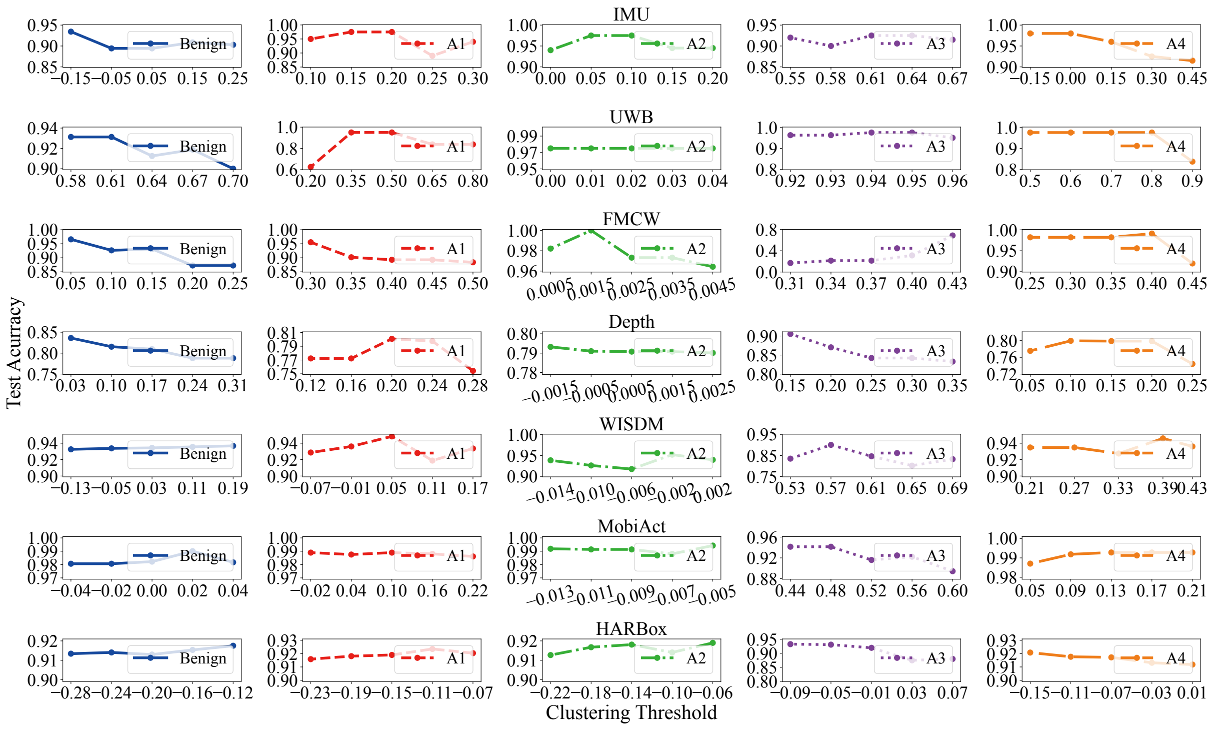


Fig. 8. Comparison of the effect of different values of clustering threshold on the final average test accuracy.

7.9 Impact of Regularization Coefficient λ

In this section, we use three large-scale datasets (i.e., WISDM, MobiAct, and HARBox) to compare the impact of different values of the regularization factor λ on the performance of FedCHAR and Ditto in benign and attack scenarios. Specifically, the value of λ is set to 0.05, 0.1, 0.5, 1, and 2, respectively, to indicate that the user-personalized model approximates the global model to different degrees.

As can be seen from Table 8, for the WISDM dataset, in the scenario where the attack type is A3, the performance of FedCHAR and Ditto is higher at $\lambda = 0.05$ compared to the other values of λ due to the more significant impact of the A3 attack on the user model training. A plausible explanation is that by reducing the value of λ to reduce the reference of the user-personalized model to the global model, the user-personalized model training is less disturbed by the global model under attack. In the scenario where the attack type is A4, the performance of FedCHAR is higher at $\lambda = 2$ compared to other values of λ. Conversely, at $\lambda = 0.05$, the performance of Ditto is higher than the other values of λ. The reason is that FedCHAR already isolates the benign nodes from the malicious nodes through clustering (see Fig. 7), so there is no need to reduce the value of λ to mitigate the impact of the attack. Increasing the value of λ to increase the reference of the user-personalized model to the global model allows the user-personalized model to learn more global features. For the MobiAct dataset, the impact of the λ value is similar to that of the WISDM dataset. For the HARBox dataset, FedCHAR always achieves better model performance than other values of λ in almost all scenarios when λ is equal to 1.

In summary, in benign scenarios and attack scenarios other than A3, FedCHAR achieves satisfactory model performance in most cases when $\lambda = 1$ is chosen. For attack types with a greater degree of attack (e.g., A3), an appropriate reduction in λ helps to prevent the user-personalized model from being affected by the attacked

Table 8. Comparison of the performance of FedCHAR and Ditto on three large-scale datasets using different values of λ.

Dataset	Method	Lambda	Benign	A1(50%)	A2(50%)	A3(50%)	A4(50%)
WISDM	FedCHAR	0.05	0.927(2.53e-3)	0.921(4.57e-3)	0.900(1.94e-2)	0.932(2.44e-3)	0.932(2.89e-3)
		0.1	0.920(4.00e-3)	0.913(4.12e-3)	0.882(7.41e-3)	**0.934(3.13e-3)**	0.932(2.79e-3)
		0.5	0.934(3.02e-3)	0.929(3.23e-3)	0.936(3.33e-3)	0.915(4.29e-3)	0.946(2.52e-3)
		1	**0.948(1.88e-3)**	**0.948(2.11e-3)**	**0.952(1.89e-3)**	0.900(2.37e-3)	0.946(3.31e-3)
		2	0.936(1.80e-3)	0.934(1.59e-3)	0.931(2.93e-3)	0.878(2.92e-3)	**0.951(1.65e-3)**
	Ditto	0.05	0.915(4.73e-3)	0.920(5.03e-3)	0.895(6.99e-3)	0.923(4.30e-3)	0.902(7.42e-3)
		0.1	0.905(3.87e-3)	0.883(1.25e-2)	0.889(1.24e-2)	0.904(6.40e-3)	0.905(6.67e-3)
		0.5	0.916(4.33e-3)	0.926(3.34e-3)	0.935(2.46e-3)	0.841(2.17e-2)	0.871(6.04e-3)
		1	0.937(2.53e-3)	0.926(3.57e-3)	0.935(2.48e-3)	0.895(3.67e-3)	0.818(1.32e-2)
		2	0.934(2.38e-3)	0.910(4.16e-3)	0.910(6.83e-3)	0.881(5.00e-3)	0.740(2.65e-2)
MobiAct	FedCHAR	0.05	0.977(1.27e-3)	0.968(2.30e-3)	0.980(2.17e-3)	**0.978(5.57e-4)**	0.985(9.94e-4)
		0.1	0.958(5.78e-3)	0.983(5.09e-4)	0.977(2.08e-3)	0.960(1.77e-3)	0.986(1.59e-3)
		0.5	0.983(9.99e-4)	0.986(7.25e-4)	0.989(9.15e-4)	0.956(2.63e-3)	0.993(3.76e-4)
		1	**0.990(3.34e-4)**	**0.994(2.09e-4)**	**0.994(2.03e-4)**	0.953(2.17e-3)	**0.994(7.08e-4)**
		2	0.985(5.36e-4)	0.990(6.28e-4)	0.991(5.17e-4)	0.873(2.99e-3)	0.993(3.19e-4)
	Ditto	0.05	0.973(2.18e-3)	0.976(1.21e-3)	0.976(1.10e-3)	0.929(8.73e-3)	0.979(1.29e-3)
		0.1	0.975(1.12e-3)	0.987(6.69e-4)	0.979(8.91e-4)	0.965(1.10e-3)	0.968(1.50e-3)
		0.5	0.986(6.48e-4)	0.985(4.59e-4)	0.987(5.53e-4)	0.954(1.81e-3)	0.928(8.04e-3)
		1	0.989(3.72e-4)	0.981(8,32e-4)	0.987(6.43e-4)	0.932(1.69e-3)	0.907(2.10e-2)
		2	0.989(3.97e-4	0.981(7.31e-4)	0.967(5.62e-3)	0.881(1.81e-3)	0.848(2.47e-2)
HARBox	FedCHAR	0.05	0.891(9.82e-3)	0.901(9.83e-3)	0.907(8.55e-3)	0.903(9.20e-3)	0.906(7.80e-3)
		0.1	0.894(9.94e-3)	0.903(9.39e-3)	0.909(7.97e-3)	0.907(1.00e-2)	0.909(9.76e-3)
		0.5	0.909(8.61e-3)	0.910(8.96e-3)	0.913(7.70e-3)	0.921(8.96e-3)	0.917(7.65e-3)
		1	**0.917(7.59e-3)**	**0.927(8.08e-3)**	**0.919(7.68e-3)**	0.932(7.52e-3)	**0.921(7.58e-3)**
		2	0.910(9.18e-3)	0.908(8.36e-3)	0.915(6.82e-3)	**0.932(6.70e-3)**	0.919(9.54e-3)
	Ditto	0.05	0.893(9.68e-3)	0.903(9.40e-3)	0.904(9.10e-3)	0.903(1.02e-2)	0.814(2.51e-2)
		0.1	0.894(9.54e-3)	0.903(1.14e-2)	0.904(8.95e-3)	0.908(9.33e-3)	0.789(2.31e-2)
		0.5	0.904(1.04e-2)	0.908(9.59e-3)	0.914(7.34e-3)	0.923(9.25e-3)	0.761(2.53e-2)
		1	0.909(9.27e-3)	0.914(9.24e-3)	0.911(8.70e-3)	0.929(8.98e-3)	0.748(3.23e-2)
		2	0.913(8.54e-3)	0.916(8.66e-3)	0.910(8.49e-3)	0.926(8.83e-3)	0.733(3.42e-2)

global model. Choosing an appropriate λ facilitates weighing the differences between the user-personalized and global models. In other experiments, by default, we set the value of λ to 1.

7.10 Scalability

In this section, we use three large-scale datasets (i.e., WISDM, MobiAct, and HARBox) to compare the performance of our approach with four other baselines (i.e., FedAvg, Ditto, Fine-tuning, and L2SGD) for different participant ratios (20%, 50%, and 80%). To ensure a balanced experiment, ClusterFL is not included in this experiment, as ClusterFL requires all users to participate in each round of training. To further validate the scalability of our approach, we combine FedCHAR with dynamic clustering and a new user mechanism to propose a new scalable framework for large-scale user training called FedCHAR-DC and conduct extensive experiments to validate its performance. Unlike FedCHAR, FedCHAR-DC requires only a fraction of users to participate in clustering and incorporates a new user mechanism to assign clusters to users dynamically. As seen from Table 9, FedCHAR-DC not only resists all types of attacks in most cases but also ensures high accuracy of user models and fairness of the model performance among users in both benign and attack scenarios.

Table 9. Comparison of the performance of FedCHAR-DC on three large-scale datasets with the other four baselines under different proportions of participants in benign and attack scenarios.

Methods	Datasets	Participants	Benign	A1(50%)	A2(50%)	A3(50%)	A4(50%)
Ditto	WISDM	20%	0.903(6.73e-3)	0.876(8.13e-3)	0.878(1.16e-2)	0.809(2.33e-2)	0.716(1.97e-2)
		50%	0.937(2.53e-3)	0.926(3.57e-3)	0.935(2.48e-3)	**0.895(3.67e-3)**	0.818(1.32e-2)
		80%	0.945(2.01e-3)	0.930(2.80e-3)	0.940(2.15e-3)	0.857(8.74e-3)	0.788(2.54e-2)
	MobiAct	20%	0.975(1.99e-3)	0.968(6.91e-3)	0.983(1.14e-3)	0.888(2.21e-3)	0.858(2.19e-2)
		50%	0.989(3.72e-4)	0.981(8.32e-4)	0.987(6.43e-3)	0.932(1.69e-3)	0.907(2.10e-2)
		80%	**0.991(2.75e-4)**	0.991(2.88e-4)	0.988(4.14e-4)	**0.965(1.35e-3)**	0.885(2.36e-2)
	HARBox	20%	0.900(9.35e-3)	0.904(1.01e-2)	0.907(1.04e-2)	0.917(9.03e-3)	0.645(3.16e-2)
		50%	0.909(9.27e-3)	0.914(9.24e-3)	0.911(8.70e-3)	0.929(8.98e-3)	0.748(3.23e-2)
		80%	0.913(7.58e-3)	0.921(6.95e-3)	0.915(8.28e-3)	0.936(7.68e-3)	0.803(2.54e-2)
FedAvg	WISDM	20%	0.800(1.06e-2)	0.780(6.59e-3)	0.841(2.91e-3)	0.383(1.18e-2)	0.234(3.72e-2)
		50%	0.839(1.06e-2)	0.736(2.50e-2)	0.806(1.95e-2)	0.429(1.17e-2)	0.386(1.28e-2)
		80%	0.842(1.06e-2)	0.825(4.97e-3)	0.862(4.37e-3)	0.707(1.04e-2)	0.353(4.24e-2)
	MobiAct	20%	0.909(1.46e-2)	0.851(2.10e-2)	0.955(8.20e-3)	0.484(2.60e-2)	0.104(9.91e-3)
		50%	0.947(7.12e-3)	0.753(4.81e-2)	0.920(1.60e-2)	0.390(1.57e-2)	0.077(4.66e-3)
		80%	0.946(5.56e-3)	0.918(6.45e-3)	0.964(1.44e-3)	0.427(9.97e-3)	0.447(2.15e-2)
	HARBox	20%	0.674(3.41e-2)	0.649(2.56e-2)	0.689(3.36e-2)	0.392(2.76e-2)	0.113(1.29e-2)
		50%	0.694(3.35e-2)	0.678(2.08e-2)	0.707(2.59e-2)	0.573(2.86e-2)	0.143(1.70e-2)
		80%	0.696(3.28e-2)	0.668(2.03e-2)	0.703(2.89e-2)	0.631(3.84e-2)	0.204(2.00e-2)
Fine-tuning	WISDM	20%	0.760(2.62e-2)	0.450(1.82e-2)	0.744(3.23e-2)	0.692(1.55e-2)	0.370(2.60e-2)
		50%	0.801(1.15e-2)	0.701(1.99e-2)	0.815(6.46e-3)	0.331(1.82e-2)	0.419(1.01e-2)
		80%	0.814(6.21e-3)	0.785(5.69e-3)	0.835(4.96e-3)	0.599(1.61e-2)	0.237(3.03e-2)
	MobiAct	20%	0.844(2.90e-2)	0.858(1.82e-2)	0.876(1.32e-2)	0.738(4.16e-2)	0.711(3.64e-2)
		50%	0.919(1.52e-2)	0.881(8.66e-3)	0.920(9.26e-3)	0.780(1.06e-2)	0.592(3.60e-2)
		80%	0.933(5.35e-3)	0.868(1.17e-2)	0.917(7.99e-3)	0.226(2.26e-2)	0.398(7.13e-2)
	HARBox	20%	0.643(3.78e-2)	0.626(2.50e-2)	0.653(2.80e-2)	0.473(3.56e-2)	0.248(2.07e-2)
		50%	0.656(3.15e-2)	0.613(2.77e-2)	0.656(2.76e-2)	0.447(1.76e-2)	0.213(2.00e-2)
		80%	0.653(3.26e-2)	0.619(2.67e-2)	0.658(2.98e-2)	0.560(2.09e-2)	0.254(2.43e-2)
L2SGD	WISDM	20%	0.851(1.08e-2)	0.831(1.38e-2)	0.833(9.67e-3)	0.825(1.08e-2)	0.825(1.13e-2)
		50%	0.895(1.29e-2)	0.883(6.47e-3)	0.873(1.45e-2)	0.316(1.83e-2)	0.870(7.34e-3)
		80%	0.867(1.61e-2)	0.884(9.60e-3)	0.886(4.28e-3)	0.391(1.32e-2)	0.828(1.75e-2)
	MobiAct	20%	0.919(9.12e-3)	0.917(9.74e-3)	0.906(1.81e-2)	0.904(1.52e-2)	0.901(2.62e-2)
		50%	0.927(1.18e-2)	0.964(1.58e-3)	0.947(6.90e-3)	0.409(1.16e-3)	0.938(9.68e-3)
		80%	0.954(2.88e-3)	0.967(2.01e-3)	0.962(4.50e-3)	0.405(1.13e-3)	0.905(1.99e-2)
	HARBox	20%	0.870(1.03e-2)	0.883(9.49e-3)	0.883(9.97e-3)	0.883(9.98e-3)	0.862(1.29e-2)
		50%	0.868(1.14e-2)	0.878(9.84e-3)	0.884(1.10e-2)	0.197(5.63e-3)	0.848(1.54e-2)
		80%	0.862(1.17e-2)	0.882(9.56e-3)	0.882(8.63e-3)	0.197(5.29e-3)	0.830(1.49e-2)
FedCHAR-DC	WISDM	20%	**0.920(6.02e-3)**	**0.895(6.83e-3)**	**0.911(6.52e-3)**	**0.836(1.28e-2)**	**0.850(1.55e-2)**
		50%	**0.937(1.92e-3)**	**0.931(3.26e-3)**	**0.943(2.42e-3)**	0.871(4.59e-3)	**0.898(6.07e-3)**
		80%	**0.949(1.27e-3)**	**0.943(2.36e-3)**	**0.952(1.92e-3)**	**0.903(2.97e-3)**	**0.921(5.87e-3)**
	MobiAct	20%	**0.981(9.39e-4)**	**0.982(7.55e-4)**	**0.987(6.63e-4)**	**0.895(1.22e-3)**	**0.975(2.40e-3)**
		50%	**0.990(3.92e-4)**	**0.988(4.96e-4)**	**0.992(3.47e-4)**	**0.938(2.59e-3)**	**0.983(8.71e-4)**
		80%	0.990(4.01e-4)	**0.994(3.22e-4)**	**0.994(2.92e-4)**	0.958(1.36e-3)	**0.973(1.61e-3)**
	HARBox	20%	**0.912(8.87e-3)**	**0.910(1.03e-2)**	**0.917(9.16e-3)**	**0.917(8.99e-3)**	**0.901(9.35e-3)**
		50%	**0.924(8.68e-3)**	**0.927(8.87e-3)**	**0.920(7.47e-3)**	**0.935(8.19e-3)**	**0.854(1.47e-2)**
		80%	**0.931(7.37e-3)**	**0.932(5.72e-3)**	**0.926(7.59e-3)**	**0.938(7.01e-3)**	**0.854(1.71e-2)**

8 LIMITATIONS AND FUTURE WORKS

In this section, we discuss the limitations of the proposed method and provide possible solutions and future research directions.

System Heterogeneity. Our work mainly focuses on solving the user data heterogeneity and its fairness problem in FL-based HAR scenarios through clustering and personalization. However, the user-device heterogeneity problem (i.e., system heterogeneity) has not been explored yet. For example, due to the different conditions of user devices, certain low-end or real-time resource-constrained devices may be struggling to train the given models and be consistently dropped out of aggregation subsequently, which also raises a fairness problem (i.e., participation fairness). In future work, we will design mechanisms that enable the server to select participating users based on the available resources of user devices. For device constraints, we can dynamically assign models with different structures or sizes to speed up the training process according to the limitations of the user devices.

Privacy of Model Updates. This work verifies through analysis and experiments where clustering can inherently resist poisoning attacks (e.g., model poisoning) to some extent. However, model updates may also be analyzed by malicious attackers or honest but curious servers (e.g., membership inference attacks, gradient inversion attacks, etc.). In future work on FL-based HAR, we will investigate how to infer information related to user activities from model updates and design corresponding defense mechanisms.

Real-world Deployment. In this work, we propose a scalable and adaptive FL framework FedCHAR-DC and validate its scalability on a single machine. The algorithmic process is centered on server clustering and user-personalized model training, so we do not consider the setting in which the server communicates with the users (i.e., across machines). In the future, we will deploy multiple devices to simulate a more realistic FL setting.

9 CONCLUSION

In this paper, we proposed FedCHAR, a clustering-based personalized federated learning framework for robust and fair HAR, which not only improves the accuracy and the fairness of model performance by combining clustering and intra-cluster personalization but also effectively improves the robustness of the FL system by isolated benign nodes from malicious nodes through clustering. In addition, to enhance the scalability of FedCHAR, we also developed a scalable and adaptive FL framework FedCHAR-DC, which is featured by dynamic clustering and adapting to the new users joining or datasets evolving for realistic FL-based HAR scenarios. We conducted extensive experiments to evaluate the performance of FedCHAR on seven datasets of various scales collected by different sensors in real-world scenarios. The results demonstrated that FedCHAR could obtain better performance on different datasets than the other five state-of-the-art methods in terms of accuracy, robustness, and fairness. We further validated that FedCHAR-DC can exhibit satisfactory scalability on three large-scale datasets regardless of the number of participants.

ACKNOWLEDGMENTS

This work is supported by the National Science Foundation of China under Grant No. 62072355, and Key Research and Development Program of Shaanxi Province of China (Grant No. 2022KWZ-10). This work is also supported by NSF (CNS-2105416, CNS-2107164, and CNS-2317190).

REFERENCES

[1] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. 2012. Human Activity Recognition on Smartphones Using a Multiclass Hardware-Friendly Support Vector Machine. In *Ambient Assisted Living and Home Care - 4th International Workshop, IWAAL 2012, Vitoria-Gasteiz, Spain, December 3-5, 2012. Proceedings (Lecture Notes in Computer Science, Vol. 7657)*, José Bravo, Ramón Hervás, and Marcela Rodríguez (Eds.). Springer, 216–223. https://doi.org/10.1007/978-3-642-35395-6_30

[2] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin B. Calo. 2019. Analyzing Federated Learning through an Adversarial Lens. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 634–643. http://proceedings.mlr.press/v97/bhagoji19a.html

[3] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. 2017. Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 119–129. https://proceedings.neurips.cc/paper/2017/hash/f4b9ec30ad9f68f89b29639786cb62ef-Abstract.html

[4] 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 2020, Glasgow, United Kingdom, July 19-24, 2020*. IEEE, 1–9. https://doi.org/10.1109/IJCNN48605.2020.9207469

[5] Youngjae Chang, Akhil Mathur, Anton Isopoussu, Junehwa Song, and Fahim Kawsar. 2020. A Systematic Study of Unsupervised Domain Adaptation for Robust Human-Activity Recognition. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 4, 1 (2020), 39:1–39:30. https://doi.org/10.1145/3380985

[6] Kaixuan Chen, Dalin Zhang, Lina Yao, Bin Guo, Zhiwen Yu, and Yunhao Liu. 2021. Deep Learning for Sensor-based Human Activity Recognition: Overview, Challenges, and Opportunities. *ACM Comput. Surv.* 54, 4 (2021), 77:1–77:40. https://doi.org/10.1145/3447744

[7] Ling Chen, Yi Zhang, and Liangying Peng. 2020. METIER: A Deep Multi-Task Learning Based Activity and User Recognition Model Using Wearable Sensors. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 4, 1 (2020), 5:1–5:18. https://doi.org/10.1145/3381012

[8] Xianda Chen, Yifei Xiao, Yeming Tang, Julio Fernandez-Mendoza, and Guohong Cao. 2021. ApneaDetector: Detecting Sleep Apnea with Smartwatches. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 5, 2 (2021), 59:1–59:22. https://doi.org/10.1145/3463514

[9] Diane J Cook, Miranda Strickland, and Maureen Schmitter-Edgecombe. 2022. Detecting Smartwatch-Based Behavior Change in Response to a Multi-Domain Brain Health Intervention. *ACM Transactions on Computing for Healthcare (HEALTH)* 3, 3 (2022), 1–18.

[10] D. Defays. 1977. An Efficient Algorithm for a Complete Link Method. *Comput. J.* 20, 4 (1977), 364–366. https://doi.org/10.1093/comjnl/20.4.364

[11] Salvatore Gaglio, Giuseppe Lo Re, and Marco Morana. 2015. Human Activity Recognition Process Using 3-D Posture Data. *IEEE Trans. Hum. Mach. Syst.* 45, 5 (2015), 586–597. https://doi.org/10.1109/THMS.2014.2377111

[12] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. 2020. An Efficient Framework for Clustered Federated Learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/e32cc80bf07915058ce90722ee17bb71-Abstract.html

[13] Fuqiang Gu, Mu-Huan Chung, Mark H. Chignell, Shahrokh Valaee, Baoding Zhou, and Xue Liu. 2022. A Survey on Deep Learning for Human Activity Recognition. *ACM Comput. Surv.* 54, 8 (2022), 177:1–177:34. https://doi.org/10.1145/3472290

[14] Filip Hanzely and Peter Richtárik. 2020. Federated Learning of a Mixture of Global and Local Models. *CoRR* abs/2002.05516 (2020). arXiv:2002.05516 https://arxiv.org/abs/2002.05516

[15] Weituo Hao, Mostafa El-Khamy, Jungwon Lee, Jianyi Zhang, Kevin J. Liang, Changyou Chen, and Lawrence Carin. 2021. Towards Fair Federated Learning With Zero-Shot Data Augmentation. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2021, virtual, June 19-25, 2021*. Computer Vision Foundation / IEEE, 3310–3319. https://doi.org/10.1109/CVPRW53098.2021.00369

[16] Zeou Hu, Kiarash Shaloudegi, Guojun Zhang, and Yaoliang Yu. 2020. FedMGDA+: Federated Learning meets Multi-objective Optimization. *CoRR* abs/2006.11489 (2020). arXiv:2006.11489 https://arxiv.org/abs/2006.11489

[17] Wei Huang, Tianrui Li, Dexian Wang, Shengdong Du, Junbo Zhang, and Tianqiang Huang. 2022. Fairness and accuracy in horizontal federated learning. *Inf. Sci.* 589 (2022), 170–185. https://doi.org/10.1016/j.ins.2021.12.102

[18] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista A. Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaïd Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha

Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. 2021. Advances and Open Problems in Federated Learning. *Found. Trends Mach. Learn.* 14, 1-2 (2021), 1–210. https://doi.org/10.1561/2200000083

[19] Jennifer R. Kwapisz, Gary M. Weiss, and Samuel Moore. 2010. Activity recognition using cell phone accelerometers. *SIGKDD Explor.* 12, 2 (2010), 74–82. https://doi.org/10.1145/1964897.1964918

[20] Chenglin Li, Di Niu, Bei Jiang, Xiao Zuo, and Jianming Yang. 2021. Meta-HAR: Federated Representation Learning for Human Activity Recognition. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, Jure Leskovec, Marko Grobelnik, Marc Najork, Jie Tang, and Leila Zia (Eds.). ACM / IW3C2, 912–922. https://doi.org/10.1145/3442381.3450006

[21] Tian Li, Ahmad Beirami, Maziar Sanjabi, and Virginia Smith. 2021. Tilted Empirical Risk Minimization. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021.* OpenReview.net. https://openreview.net/forum?id=K5YasWXZT3O

[22] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. 2021. Ditto: Fair and Robust Federated Learning Through Personalization. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 6357–6368. http://proceedings.mlr.press/v139/li21h.html

[23] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Process. Mag.* 37, 3 (2020), 50–60. https://doi.org/10.1109/MSP.2020.2975749

[24] Tian Li, Maziar Sanjabi, Ahmad Beirami, and Virginia Smith. 2020. Fair Resource Allocation in Federated Learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020.* OpenReview.net. https://openreview.net/forum?id=ByexElSYDr

[25] Xinyu Li, Yanyi Zhang, Ivan Marsic, Aleksandra Sarcevic, and Randall S Burd. 2016. Deep learning for rfid-based activity recognition. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM.* 164–175.

[26] Yadong Li, Dongheng Zhang, Jinbo Chen, Jinwei Wan, Dong Zhang, Yang Hu, Qibin Sun, and Yan Chen. 2021. Towards Domain-Independent and Real-Time Gesture Recognition Using mmWave Signal. *CoRR* abs/2111.06195 (2021). arXiv:2111.06195 https://arxiv.org/abs/2111.06195

[27] Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. 2020. Federated Learning in Mobile Edge Networks: A Comprehensive Survey. *IEEE Commun. Surv. Tutorials* 22, 3 (2020), 2031–2063. https://doi.org/10.1109/COMST.2020.2986024

[28] Haojie Ma, Zhijie Zhang, Wenzhong Li, and Sanglu Lu. 2021. Unsupervised Human Activity Representation Learning with Multi-task Deep Clustering. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 5, 1 (2021), 48:1–48:25. https://doi.org/10.1145/3448074

[29] Othmane Marfoq, Giovanni Neglia, Aurélien Bellet, Laetitia Kameni, and Richard Vidal. 2021. Federated Multi-Task Learning under a Mixture of Distributions. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (Eds.). 15434–15447. https://proceedings.neurips.cc/paper/2021/hash/82599a4ec94aca066873c99b4c741ed8-Abstract.html

[30] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA (Proceedings of Machine Learning Research, Vol. 54)*, Aarti Singh and Xiaojin (Jerry) Zhu (Eds.). PMLR, 1273–1282. http://proceedings.mlr.press/v54/mcmahan17a.html

[31] El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. 2018. The Hidden Vulnerability of Distributed Learning in Byzantium. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018 (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer G. Dy and Andreas Krause (Eds.). PMLR, 3518–3527. http://proceedings.mlr.press/v80/mhamdi18a.html

[32] Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh. 2019. Agnostic Federated Learning. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 4615–4625. http://proceedings.mlr.press/v97/mohri19a.html

[33] Fionn Murtagh and Pedro Contreras. 2012. Algorithms for hierarchical clustering: an overview. *WIREs Data Mining Knowl. Discov.* 2, 1 (2012), 86–97. https://doi.org/10.1002/widm.53

[34] Tushar Nagarajan, Yanghao Li, Christoph Feichtenhofer, and Kristen Grauman. 2020. Ego-Topo: Environment Affordances From Egocentric Video. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020.* Computer Vision Foundation / IEEE, 160–169. https://doi.org/10.1109/CVPR42600.2020.00024

[35] Kai Niu, Fusang Zhang, Zhaoxin Chang, and Daqing Zhang. 2018. A Fresnel Diffraction Model Based Human Respiration Detection System Using COTS Wi-Fi Devices. In *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers, UbiComp/ISWC 2018 Adjunct, Singapore, October 08-12, 2018.* ACM, 416–419. https://doi.org/10.1145/3267305.3267561

[36] George A. Oguntala, Raed A. Abd-Alhameed, Nazar T. Ali, Yim-Fun Hu, James M. Noras, Nnabuike N. Eya, Issa T. E. Elfergani, and Jonathan Rodriguez. 2019. SmartWall: Novel RFID-Enabled Ambient Human Activity Recognition Using Machine Learning for

Unobtrusive Health Monitoring. *IEEE Access* 7 (2019), 68022–68033. https://doi.org/10.1109/ACCESS.2019.2917125

[37] Xiaomin Ouyang, Zhiyuan Xie, Jiayu Zhou, Jianwei Huang, and Guoliang Xing. 2021. ClusterFL: a similarity-aware federated learning system for human activity recognition. In *MobiSys '21: The 19th Annual International Conference on Mobile Systems, Applications, and Services, Virtual Event, Wisconsin, USA, 24 June - 2 July, 2021*, Suman Banerjee, Luca Mottola, and Xia Zhou (Eds.). ACM, 54–66. https://doi.org/10.1145/3458864.3467681

[38] Yili Ren, Zi Wang, Sheng Tan, Yingying Chen, and Jie Yang. 2021. Winect: 3D Human Pose Tracking for Free-form Activity Using Commodity WiFi. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 5, 4 (2021), 176:1–176:29. https://doi.org/10.1145/3494973

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

[40] Robin Sibson. 1973. SLINK: An Optimally Efficient Algorithm for the Single-Link Cluster Method. *Comput. J.* 16, 1 (1973), 30–34. https://doi.org/10.1093/comjnl/16.1.30

[41] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet Talwalkar. 2017. Federated Multi-Task Learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 4424–4434. https://proceedings.neurips.cc/paper/2017/hash/6211080fa89981f66b1a0c9d55c61d0f-Abstract.html

[42] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H. Brendan McMahan. 2019. Can You Really Backdoor Federated Learning? *CoRR* abs/1911.07963 (2019). arXiv:1911.07963 http://arxiv.org/abs/1911.07963

[43] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. 2020. Data Poisoning Attacks Against Federated Learning Systems. In *Computer Security - ESORICS 2020 - 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14-18, 2020, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 12308)*, Liqun Chen, Ninghui Li, Kaitai Liang, and Steve A. Schneider (Eds.). Springer, 480–501. https://doi.org/10.1007/978-3-030-58951-6_24

[44] Linlin Tu, Xiaomin Ouyang, Jiayu Zhou, Yuze He, and Guoliang Xing. 2021. FedDL: Federated Learning via Dynamic Layer Sharing for Human Activity Recognition. In *SenSys '21: The 19th ACM Conference on Embedded Networked Sensor Systems, Coimbra, Portugal, November 15 - 17, 2021.* ACM, 15–28. https://doi.org/10.1145/3485730.3485946

[45] George Vavoulas, Charikleia Chatzaki, Thodoris Malliotakis, Matthew Pediaditis, and Manolis Tsiknakis. 2016. The MobiAct Dataset: Recognition of Activities of Daily Living using Smartphones. In *Proceedings of the 2nd International Conference on Information and Communication Technologies for Ageing Well and e-Health, ICT4AgeingWell 2016, Rome, Italy, April 21-22, 2016*, Carsten Röcker, Martina Ziefle, John O'Donoghue, Leszek A. Maciaszek, and William Molloy (Eds.). SCITEPRESS, 143–151. https://doi.org/10.5220/0005792401430151

[46] Jindong Wang, Yiqiang Chen, Shuji Hao, Xiaohui Peng, and Lisha Hu. 2019. Deep learning for sensor-based activity recognition: A survey. *Pattern Recognit. Lett.* 119 (2019), 3–11. https://doi.org/10.1016/j.patrec.2018.02.010

[47] Lei Wang, Xiang Zhang, Yuanshuang Jiang, Yong Zhang, Chenren Xu, Ruiyang Gao, and Daqing Zhang. 2021. Watching Your Phone's Back: Gesture Recognition by Sensing Acoustical Structure-borne Propagation. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 5, 2 (2021), 82:1–82:26. https://doi.org/10.1145/3463522

[48] Zheng Wang, Xiaoliang Fan, Jianzhong Qi, Chenglu Wen, Cheng Wang, and Rongshan Yu. 2021. Federated Learning with Fair Averaging. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, Zhi-Hua Zhou (Ed.). ijcai.org, 1615–1623. https://doi.org/10.24963/ijcai.2021/223

[49] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. 2019. Fall of Empires: Breaking Byzantine-tolerant SGD by Inner Product Manipulation. In *Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI 2019, Tel Aviv, Israel, July 22-25, 2019 (Proceedings of Machine Learning Research, Vol. 115)*, Amir Globerson and Ricardo Silva (Eds.). AUAI Press, 261–270. http://proceedings.mlr.press/v115/xie20a.html

[50] Xinyi Xu and Lingjuan Lyu. 2020. A Reputation Mechanism Is All You Need: Collaborative Fairness and Adversarial Robustness in Federated Learning. https://doi.org/10.48550/ARXIV.2011.10464

[51] Odilia Yim and Kylee Ack Baraly. 2015. Hierarchical Cluster Analysis: Comparison of Three Linkage Measures and Application to Psychological Data. *The Quantitative Methods for Psychology* 11 (02 2015), 8–21. https://doi.org/10.20982/tqmp.11.1.p008

[52] Dong Yin, Yudong Chen, Kannan Ramchandran, and Peter L. Bartlett. 2018. Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018 (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer G. Dy and Andreas Krause (Eds.). PMLR, 5636–5645. http://proceedings.mlr.press/v80/yin18a.html

[53] Tao Yu, Eugene Bagdasaryan, and Vitaly Shmatikov. 2020. Salvaging Federated Learning by Local Adaptation. *CoRR* abs/2002.04758 (2020). arXiv:2002.04758 https://arxiv.org/abs/2002.04758

[54] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. 2018. Federated Learning with Non-IID Data. *CoRR* abs/1806.00582 (2018). arXiv:1806.00582 http://arxiv.org/abs/1806.00582