EchoPFL: Asynchronous Personalized Federated Learning on Mobile Devices with On-Demand Staleness Control

XIAOCHEN LI, Northwestern Polytechnical University, China
SICONG LIU, Northwestern Polytechnical University, China
ZIMU ZHOU, City University of Hong Kong, China
BIN GUO, Northwestern Polytechnical University, China
YUAN XU, Northwestern Polytechnical University, China
ZHIWEN YU, Harbin Engineering University, China and Northwestern Polytechnical University, China

The rise of mobile devices with abundant sensory data and local computing capabilities has driven the trend of federated learning (FL) on these devices. And personalized FL (PFL) emerges to train specific deep models for each mobile device to address data heterogeneity and varying performance preferences. However, mobile training times vary significantly, resulting in either delay (when waiting for slower devices for aggregation) or accuracy decline (when aggregation proceeds without waiting). In response, we propose a shift towards asynchronous PFL, where the server aggregates updates as soon as they are available. Nevertheless, existing asynchronous protocols are unfit for PFL because they are devised for federated training of a single global model. They suffer from slow convergence and decreased accuracy when confronted with severe data heterogeneity prevalent in PFL. Furthermore, they often exclude slower devices for staleness control, which notably compromises accuracy when these devices possess critical personalized data. Therefore, we propose EchoPFL, a coordination mechanism for asynchronous PFL. Central to EchoPFL is to include updates from all mobile devices regardless of their latency. To cope with the inevitable staleness from slow devices, EchoPFL revisits model broadcasting. It intelligently converts the unscalable broadcast to *on-demand broadcast*, leveraging the *asymmetrical bandwidth* in wireless networks and the dynamic clustering-based PFL. Experiments show that compared to status quo approaches, EchoPFL achieves a reduction of up to 88.2% in convergence time, an improvement of up to 46% in accuracy, and a decrease of 37% in communication costs.

CCS Concepts: • **Human-centered computing** → **Ubiquitous and mobile computing**; • **Computing methodologies** → **Machine learning**.

Additional Key Words and Phrases: Asynchronous personalized federated learning, data heterogeneity, dynamic clustering, on-demand broadcast

ACM Reference Format:
Xiaochen Li, Sicong Liu, Zimu Zhou, Bin Guo, Yuan Xu, and Zhiwen Yu. 2024. EchoPFL: Asynchronous Personalized Federated Learning on Mobile Devices with On-Demand Staleness Control. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 8, 1, Article 41 (January 2024), 23 pages. https://doi.org/10.1145/3643560

Corresponding author: scliu@nwpu.edu.cn.
Authors' addresses: Xiaochen Li, Northwestern Polytechnical University, School of Computer Science, Xi'an, China; Sicong Liu, Northwestern Polytechnical University, School of Computer Science, Xi'an, China ; Zimu Zhou, City University of Hong Kong, School of Data Science, Hong Kong, China; Bin Guo, Northwestern Polytechnical University, School of Computer Science, Xi'an, China; Yuan Xu, Northwestern Polytechnical University, School of Computer Science, Xi'an, China; Zhiwen Yu, Harbin Engineering University, Harbin, China and Northwestern Polytechnical University, School of Computer Science, Xi'an, China.

2474-9567/2024/1-ART41 $15.00
https://doi.org/10.1145/3643560

1 INTRODUCTION

The rapid growth of sensory data generated from ubiquitous mobile devices, coupled with their local computing power, in addition to the widespread availability of wireless networks, has catalyzed the emergence of federated learning (FL) on these devices. In this paradigm, multiple *clients*, *i.e.*, ubiquitous mobile devices such as smartphones, wearables, drones, and robots, collaboratively train a shared model in a specific application scenario under *server* orchestration while keeping their datasets decentralized [28, 29, 39]. FL offers an avenue for the development of data-intensive deep learning applications with ubiquitous mobile devices, including activity recognition [33, 44], personalized recommendation [34, 42, 56], and transportation [20, 35, 57]. Attributed to diverse user behaviors and preferences, sensory data from ubiquitous mobile devices are often non-IID (identically and independently distributed). For example, one user prefers outdoor activities, while another prefers indoor hobbies. Their sensory data, such as GPS location or activity trackers, would exhibit distinct patterns. This makes it challenging to learn a single model for all clients with high accuracy [7, 54].

To handle such natural *data heterogeneity* on ubiquitous mobile devices, *personalized federated learning (PFL)* has been introduced. PFL seeks to train client-distinct models to accommodate the diverse data distributions across different clients [54]. PFL strategies roughly fall into *global model personalization* or *learning personalized models*. The former includes techniques such as local fine-tuning [59, 66, 68] and meta-learning [16], while the latter embraces methods like clustering [4, 17, 33, 44], multi-task learning [50], and knowledge distillation [67].

However, there exists a significant *gap* when it comes to the practical deployment of such PFL systems in real-world ubiquitous mobile application scenarios. The gap is the variations in local training time, especially due to the diverse computing resources and network availability of different devices. Most PFL frameworks, such as those proposed in studies like [4, 17, 19, 44, 50, 67, 72], are developed with the assumption of *synchronous* model aggregation, where the server waits for updates from all clients in each round. In essence, *slow devices* can induce significant waiting time and thus the training delay. However, collaborative training with mobile devices is always latency-sensitive in ubiquitous applications, for example [10, 70]. Compounding this issue, the specialized designs in PFL often come with even higher computation and communication costs than the non-personalized counterparts [7]. This brings us to a potential solution: *asynchronous PFL*, where the server aggregates updates when they arrive from clients, eliminating the waiting time associated with stragglers. In particular, many alternative solutions exist to address the challenge of *mobile system heterogeneity*, such as client selection [28, 30, 41], adaptive learning rate control [64], and heterogeneous model architectures [12, 13]. Among them, the asynchronous strategy is notable, offering simplified client-server coordination and enhanced adaptability to mobile device resource fluctuations [65].

Nevertheless, asynchronous PFL with mobile devices introduces its own set of challenges. Asynchronous protocols might incur excessive communication overhead and degraded accuracy due to model *staleness* [40, 45, 63]. To balance model accuracy and training latency, semi-asynchronous FL [38, 53, 61] has been introduced, where clients synchronize with the server at carefully controlled frequencies. However, we note that these protocols are primarily designed for training a single model and may encounter challenges when applied in the context of PFL (see Sec. 2). Also, researchers [15, 65] report slow convergence and considerable accuracy drop of asynchronous protocols with severe data heterogeneity—an issue that is present in PFL. A more notable concern is the potential exclusion of slower devices for staleness control [61]. When the slow devices contain large amounts of important personalized data, excluding them from training would drastically deteriorate the model accuracy [53].

In this paper, we propose EchoPFL, a simple yet effective client-server coordination mechanism via *proactive on-demand model broadcast* for staleness control in asynchronous PFL with mobile devices. At its core, EchoPFL ensures no critical data is left behind by including updates from all devices, regardless of their local training latency. Specifically, to manage the inevitable model staleness from slower devices, the server timely broadcasts the most recent aggregated models to clients involved in the training of the same personalized cluster, akin to an

"echo". Note that proactive broadcast is seldom applied in prior asynchronous FL systems because it introduces excessive server-client communication. Instead, we find broadcasts suited for asynchronous PFL with mobile devices based on the following ***novel observations***:

- Wireless networks connecting ubiquitous devices often feature *asymmetric bandwidth*. For example, the downstream bandwidth can be up to 10× larger than the upstream bandwidth, in a typical 5G network [8]. This *asymmetry* is particularly advantageous because the model broadcast from the server exclusively utilizes the downstream traffic, preventing network congestion and thus long latency.
- The data heterogeneity in PFL can be seen as *blessing* rather than a burden with the model broadcasting. This is because data heterogeneity allows for a more targeted approach to broadcasting model updates. By confining model broadcasts to clients with similar data distributions, the scope of the broadcast is constrained. Moreover, since broadcast takes place on relatively homogeneous data, it also increases the tolerance to staleness and potentially reduces the broadcast frequency [44].

Specifically, the design of EchoPFL is grounded upon clustering-based PFL frameworks [4, 17, 44], as many datasets in mobile computing applications exhibit high clusterability [44]. To rapidly and accurately assign clients to appropriate clusters based on asynchronously arrived information, EchoPFL adopts data-aware dynamic client clustering to incrementally create and manage clusters. Moreover, EchoPFL periodically merges/expands clusters in line with potential drifts in mobile client data. Within each cluster, EchoPFL also predicts the optimal broadcast frequency to further reduce the downstream communication cost without compromising accuracy. We implement EchoPFL as a continuous integration (CI) based client-server coordination scheme which makes it promising for integration with mainstream FL frameworks, such as FLOWER [2]. We evaluate the performance of EchoPFL on four mobile tasks and four real-world scenarios with diverse data or system heterogeneity using twenty mobile devices. Results show a reduction of up to 88.2% in training time and up to 37% in communication cost with an improvement of up to 41.04% in accuracy. Especially, EchoPFL achieves up to 46% accuracy increase in slow clients (Sec. 7). Our main contributions are summarized as follows.

- To the best of our knowledge, this is the first work that effectively integrates ubiquitous system asynchrony into personalized FL. It not only ensures the inclusion of slower mobile clients but also effectively tackles the challenge of model staleness without compromising any mobile model accuracy.
- We propose EchoPFL, asynchronous personalized FL with mobile devices via on-demand model broadcast. It harnesses the data heterogeneity in personalized FL and the bandwidth asymmetry in wireless networks via data-aware dynamic client clustering and in-cluster adaptive model broadcast. We also implement EchoPFL as an easy-to-use client-server coordination scheme for integration with other FL frameworks.
- Experiments show that EchoPFL outperforms existing a-/semi-/synchronous or personalized FL methods [28, 39, 44, 63] in trading off between accuracy and training time at low communication costs across various mobile tasks, platforms, and scenarios. It also yields a significant accuracy increase for slow devices.

2 MOTIVATION AND CHALLENGE

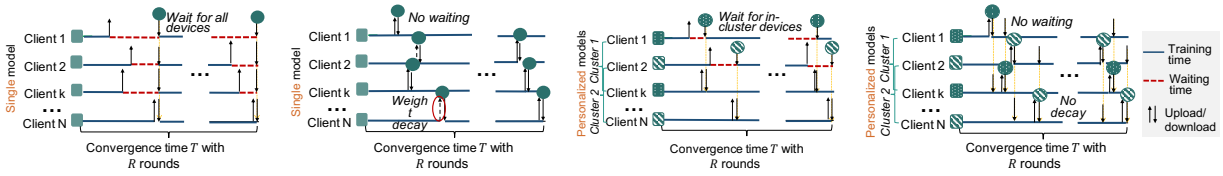
In this section, we delve into the potential advantages and challenges of asynchronous PFL, drawing insights from preliminary studies on two mobile applications.

2.1 Motivation for Asynchronous Personalized FL

In the context of mobile applications, FL systems aim to learn *accurate* deep models with *low latency*, allowing for fast adaptation to dynamic contexts and user preferences. Generally, Federated Learning (FL) constitutes a collaborative training process between multiple mobile clients and a server [28, 29, 39, 58]. Mobile clients perform local training with their local datasets and subsequently upload their trained deep models to the server. The

Table 1. Performance comparisons of different FL paradigms on two widely used mobile applications.

Representative paradigms	Image recognition (IR)				Human activity recognition (HAR)			
	Accuracy (%)	Accuracy at the slowest device (%)	Accuracy at the fastest device (%)	Latency (min)	Accuracy (%)	Accuracy at the slowest device (%)	Accuracy at the fastest device (%)	Latency (min)
Sync FL (FedAvg [39])	52.1 ± 15.3	30.5	60.6	398	89.7±5.3	82.1	94.6	37.2
Async FL (FedAsyn [63])	48.6 ± 22.5	22.1	71.3	102	85.6±11.4	65.4	91.2	17.4
Sync PFL (ClusterFL [44])	92.4 ± 3.7	89.6	94.3	321	99.6±0.2	98.9	100	33.6



(a) Sync FL (b) Async FL (c) Sync PFL (d) Async PFL

Fig. 1. Potential gains in training latency of asynchronous PFL over existing FL paradigms.

server aggregates these models, often using weighted averaging, and then distributes the updated model back to the mobile clients. This iterative process continues until convergence.

However, standard FL algorithms, such as FedAvg [39], encounter performance degradation when applied to mobile devices. The reasons are two-fold:

- A unified global model yields low accuracy with non-IID sensory data across mobile devices, necessitating various *personalized* global models for individual mobile devices [7, 54].
- The primary *latency bottleneck* in each training round is the need to wait for stragglers, *e.g.*, low-speed computing devices. This highlights the importance of considering *asynchrony* from a system perspective [40, 45, 63].

Current FL paradigms with mobile devices that emphasize either *data* or *resource* heterogeneity fail to strike an optimal balance between model accuracy and training latency. To illustrate, we evaluate the performance of representative paradigms with two tasks: *image recognition* and *human activity recognition*. The former encompasses applications such as smartphone-based face recognition [25] and robot-driven security patrolling [14]. The latter is widely used in health-care [24, 52, 60, 69] and motion detection [73]. Tab. 1 presents the results on the CIFAR-10 [27] (for image classification) and HAR-UCI [1] (for human activity recognition) employing 12 diverse mobile devices using three FL paradigms: synchronous FL (FedAvg [39]), asynchronous FL (FedAsyn [63]), and synchronous PFL (ClusterFL [44]). The detailed experimental setups are available in Sec. 7.1. Note that our focus on clustering-based PFL algorithms arises from the high clusterability observed in the data distributions across many mobile applications [6, 44, 51]. We make the following observations.

- Synchronous personalized FL (PFL) significantly outperforms synchronous FL in ***accuracy***. In image recognition, synchronous PFL improves the accuracy by at least 40.3% over synchronous FL. For human activity recognition, it shows a 9.9% accuracy advantage over synchronous FL.
- The ***accuracy*** of synchronous personalized FL is also 4.1% higher than asynchronous FL (16.7% higher on the lowest device), yet with longer training latency.
- Asynchronous FL drastically decreases training ***latency***. In image recognition, it reduces the training time by 74.4% compared to synchronous FL, whereas in human activity recognition, the reduction is 53.2%.

These observations motivate us to not only adopt the PFL paradigm to ensure the accuracy of different ubiquitous mobile applications but also to further reduce the training latency of PFL by transitioning from synchronous to asynchronous model aggregation, *i.e.*, asynchronous PFL. The intuition is also demonstrated in Fig. 1. Specifically, asynchronous FL effectively reduces latency by eliminating the waiting time for slow devices

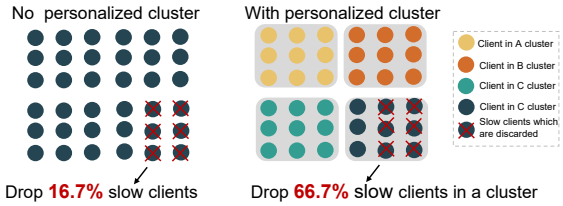
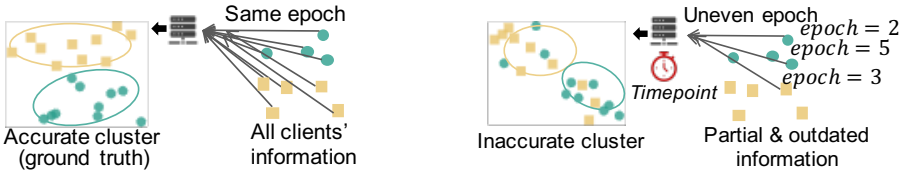


Fig. 2. Impact of slow device model dropping on PFL.



(a) Synchronous PFL (b) Asynchronous PFL

Fig. 3. Impact of asynchrony on clustering.

during model aggregation (Fig. 1b vs. Fig. 1a). Synchronous PFL (via clustering) employs personalized models to balance generalization and personalization, meanwhile accelerates the training process by only waiting for the slow devices within the same clusters (Fig. 1c vs. Fig. 1a). Notably, the training latency of synchronous PFL can be further optimized by enabling asynchronous aggregation within clusters (Fig. 1d vs. Fig. 1c). Therefore, by shifting to asynchronous PFL(APFL), we would achieve both higher model accuracy and lower training latency.

2.2 Challenges in Asynchronous PFL with Mobile Devices

While asynchronous protocols [40, 45, 63] and personalization techniques [4, 17, 19, 44, 50, 67] exist for FL, seamlessly integrating the two introduces non-trivial challenges.

Challenge #1: How to adapt the personalization strategies to the asynchronous mobile system setting? Prior PFL frameworks [44, 48] implicitly assume a synchronous, globally comprehensive information collection process for subsequent processes, *e.g.*, clustering. However, in real-world asynchronous system settings, where mobile clients may not respond to the server within the desired time limit, the server must cluster mobile clients based on partial information. This process must also operate under real-time constraints, potentially leading to less accurate clustering results. For example, as shown in Fig. 3, existing synchronous PFL methods cluster mobile clients by utilizing the same epoch to measure specific probabilistic distance metrics among the local model updates returned by all mobile clients [44, 48]. However, when local model updates inevitably arrive at the server in different epochs, this approach fails to promptly and accurately identify clusters.

Challenge #2: How to control the model staleness without decaying or discarding updates from slow mobile devices in personalized FL process? Asynchronous FL protocols often adopt weight decay [45, 63] or model dropping [38, 61] to mitigate the negative impact on aggregated model accuracy caused by outdated models. However, we argue that these methods can result in a significant degradation in the accuracy of personalized models. This is because slow devices may possess crucial data necessary for specific personalized models. To illustrate this point, consider the toy example presented in Figure 2. Excluding 6 slow devices from a conventional FL (*e.g.*, FedAvg [39]) neglects a mere 16.7% of data for the single model (marked in blue). Conversely, discarding the same 6 devices in clustering-based PFL (*e.g.*, ClusterFL [44]) leads to a pronounced data loss of 66.7% for that cluster (marked in blue). Such data omission can significantly damage the model accuracy for that cluster.

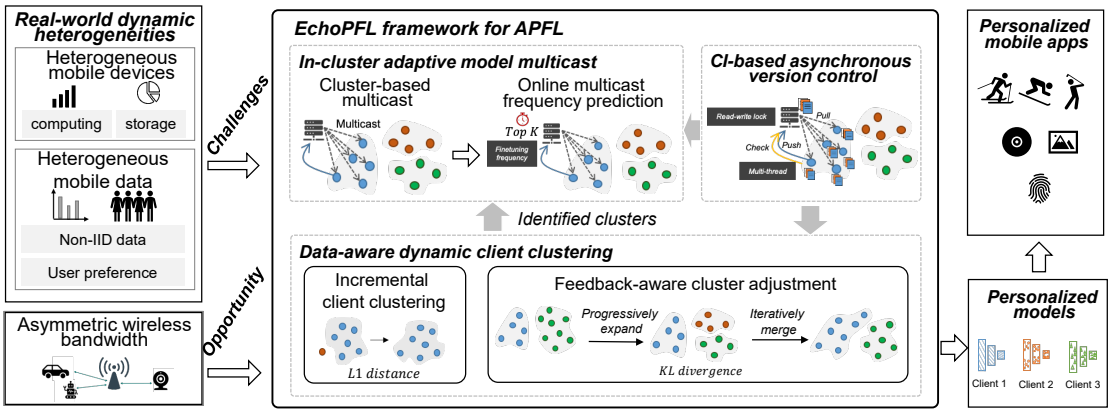


Fig. 4. Overview of EchoPFL for asynchronous PFL with mobile devices.

3 SOLUTION OVERVIEW

This section presents an overview of EchoPFL, a mobile client-server coordination mechanism for asynchronous PFL. EchoPFL resolves *Challenge #1* in Sec. 4 through *dynamic client clustering*, which remains efficient and effective despite the asynchronous arrival of mobile client information. EchoPFL addresses *Challenge #2* in Sec. 5 by revisiting *model broadcast* mechanism, previously deemed unscalable for model staleness control in FL, and refining it into *on-demand model broadcast*.

Design Rationales. As mentioned in Sec. 1, *on-demand model broadcast* becomes viable for model staleness control in PFL with mobile devices, given the inherently limited model broadcast scope in clustering-based PFL. The core principle guiding the management of model staleness in PFL is the imperative need to ensure that the model updates used in the training process remain relatively up-to-date. This necessity becomes particularly critical in light of the asynchronous nature of ubiquitous mobile systems, which can introduce delays in delivering specific mobile model updates to the server. These delays may potentially lead to the utilization of outdated models during the aggregation process and subsequent training rounds. In essence, on-demand model broadcast addresses these challenges by enabling the broader and more timely dissemination of model updates. This is a departure from the traditional asynchronous FL, where updates are typically confined to specific devices. The broader broadcast scope ensures the rapid distribution of model updates to a wider audience, thereby reducing model staleness. Consequently, it effectively transforms into an "on-demand model broadcast" problem, which is exclusive to clients sharing similar data distributions, such as those within clusters. This approach effectively addresses model staleness while minimizing communication overhead. It's worth noting that broadcast operations over homogeneous data, *i.e.*, within a cluster, can enhance the system's resilience to staleness, potentially reducing the need for frequent broadcasts [44]. Furthermore, on-demand broadcast leverages the more abundant downstream traffic rather than its upstream link. Accordingly, on-demand broadcast from the server to clients is unlikely to trigger notable network congestion and thus latency surges.

System Workflow. EchoPFL consolidates the above rationales into an *on-demand model broadcast* mechanism, exemplified within clustering-based PFL frameworks [44, 48]. Fig. 4 shows the architecture of EchoPFL. It mainly consists of two functional modules:

- *Data-aware Dynamic Client Clustering* (Sec. 4). As client clustering identifies the scope of model broadcast for staleness control, it is crucial to conduct rapid and accurate client clustering in light of asynchronously arriving local model updates. EchoPFL dynamically creates and manages clusters via on-arrival initial clustering and periodic feedback-aware cluster refinement.

- *In-cluster Adaptive Model broadcast* (Sec. 5). After clustering, EchoPFL broadcasts the latest aggregated models to in-cluster clients as required. Due to the relatively homogeneous data distribution within a cluster, training is resilient against a certain degree of staleness. Also, EchoPFL predicts the optimal broadcast frequency to further reduce the downstream communication cost without compromising accuracy.

To manage asynchronous version conflicts on server and clients, we further implement EchoPFL as a continuous integration (CI) based client-server coordination scheme for easy integration with other FL frameworks (Sec. 6).

4 DATA-AWARE DYNAMIC CLUSTERING

As mentioned in Sec. 3, rapid and accurate mobile client clustering in light of asynchronously arriving local model updates not only constitutes a critical stage in realizing PFL but also plays an essential role in defining the broadcast scope. We first discuss the requirements and challenges for client clustering in asynchronous PFL, before explaining the designs of our data-aware dynamic clustering scheme.

4.1 Primer on Mobile Client Clustering

Clustering in personalized FL aims to gather mobile clients whose local sensory datasets share similar distributions into the same cluster for training [23, 44, 48]. Since the local datasets are inaccessible in FL, the server typically measures the similarity between the models uploaded by the mobile clients as proxy to the similarity of local data distributions. The similarity between models can be measured at the *parameter* level metric (*e.g.*, directly compare model parameters via L1 distance) or at the *feature* level (*e.g.*, compare feature maps of model outputs via KL divergence). Feature-level metrics better reflect the similarity of data distributions among mobile clients than parameter-level metrics [44]. Yet the feature-level metrics suffer from longer latency because it needs to collect outputs of model inference. For *fast* and *accurate* client clustering, EchoPFL adopts a two-phase scheme by first promptly assigning mobile clients to an initial cluster upon arrival (Sec. 4.2), and then periodically refining the clusters via feedback from mobile clients (Sec. 4.3). We elaborate on the two designs below.

4.2 On-arrival Initial Cluster Assignment of Mobile Devices

This module immediately assigns a client to a cluster upon receiving its model update to achieve real-time client clustering in the asynchronous setting.

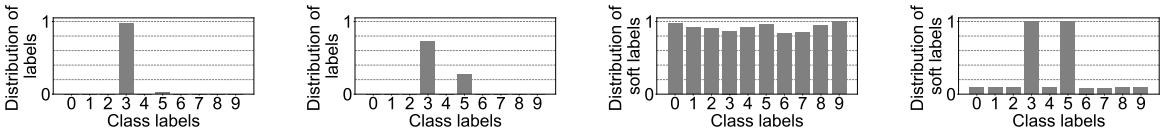
4.2.1 Mobile Cluster Initialization. Since model updates arrive asynchronously, EchoPFL initializes the clusters incrementally [5]. Given a predefined number of clusters C, EchoPFL initializes the centers of C clusters as the first C local model parameters that arrive at the server.

4.2.2 Mobile Cluster Assignment. For the newly arrived model parameters u_i from a client i, EchoPFL calculates the L1 distance L between u_i to all C clusters and assigns client i to the cluster with the smallest L1 distance:

$$\begin{aligned} cluster &= \arg\min \quad (L(u_i, v_1), L(u_i, v_2), \ldots, L(u_i, v_C)) \\ where &\quad L(u_i, v_c) = \|u_i - v_c\| \end{aligned} \tag{1}$$

u_i is the model parameters uploaded by client i, and v_c represents the model parameters of the c-th cluster center.

We adopt parameter-level similarity metrics rather than feature-level for real-time mobile client clustering. As we will show in Sec. 7.4, the feature-level similarity metrics could be $12,000\times$ slower than the parameter-level counterpart in this process. Furthermore, feature-level similarity metrics using KL divergence [11, 44] may incur considerable I/O contention. Despite its high efficiency, the initial clustering could be erroneous, which results in the feedback-aware clustering refinement, as we will discuss next.



(a) F_c, $round = 1$ (b) F_c, $round = 100$ (c) S_c, $round = 1$ (d) S_c, $round = 100$

Fig. 5. Illustration of predicted label distribution (F_c) and soft label distribution (S_c) based on mobile-side model weights at different training rounds (*i.e.*, $round = 1$ and 100) on the same sensory data distribution.

4.3 Feedback-aware Mobile Cluster Refinement

This module improves the accuracy of the initial clusters by periodical refinement, *e.g.*, merging and expanding clusters based on the mobile *client feedback*.

4.3.1 Assessing Clustering Accuracy via Mobile Client Feedback. We devise a client feedback scheme to assess the clustering accuracy. The feedback $g(v_c, \Pi_i)$ of client i measures how the model parameters v_c from its assigned cluster c fits its local dataset Π_i. Specifically, client i performs inference using v_c on Π_i and records the distribution F_c of the predicted labels. It then compares F_c with the distribution F_i of the actual labels and measures their difference via the chi-squared test $\mathcal{X}^2()$. That is, the feedback of mobile client i is calculated as:

$$g(v_c, \Pi_i) = \mathcal{X}^2(F_c, F_i) = \sum_{j=1}^{J} \frac{(F_c^j - F_i^j)^2}{F_i^j} \tag{2}$$

where F_c^j is the observed frequency of the j-th class, F_i^j is the expected frequency of the j-th class, and J is the number of classes. As a separate note, we employ the chi-squared test here to assess distribution differences and account for their non-IID nature and discreteness [3].

However, the client feedback calculated by $\mathcal{X}^2(F_c, F_i)$ reflects not only the clustering accuracy but also the PFL training stage. Accordingly, we need to separate and remove the impact of PFL training stages, so that the client feedback only precisely indicates the accuracy of clustering. This is non-trivial because the early- and late-stage trained models from slow and fast devices will disturb each other. We observe that the distribution of predicted labels F_c by $c-th$ cluster center model exhibits distinct patterns during different training stages, eliminating the impact of training from the client feedback. Fig. 5 shows the predicted label distribution F_c and the predicted *soft label* distribution S_c at different training stages ($round = 1$ or 100). The soft labels are class probabilities produced by the $c-th$ cluster center model's weights. It reveals two observations:

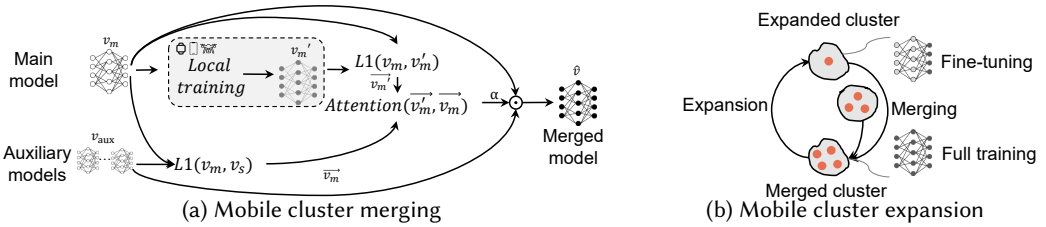
- The consistency in predicted label distributions shows the viability of employing the chi-squared test to assess label distributions, as highlighted in Fig. 5a and Fig. 5b.
- The predicted distribution of soft labels at various training stages vary, as shown Fig. 5c and Fig. 5d.

In summary, the model parameters v_c from the $c-th$ cluster manifest more significant variances in soft label distributions for predictions across different categories. Accordingly, it could be used as a proxy for the training stage, and thus rectify the errors in client feedback. Specifically, we introduce a probabilistic variance as a measure of *training sufficiency*, and revise the client feedback calculation as follows:

$$g(v_c, \Pi_i) \approx \sum_{j=1}^{J} \frac{(F_c^j - F_i^j)^2}{F_i^j} \cdot \mathrm{Var}(S_c) \tag{3}$$

where $Var(S_c)$ represents the variance of the predicted distribution of soft labels.

4.3.2 Mobile Cluster Merging. As the model parameters of different cluster centers are trained separately for an extended duration, they fit diverse and potentially conflicting local data distributions. Naive aggregation



(a) Mobile cluster merging (b) Mobile cluster expansion

Fig. 6. Illustration of mobile cluster expansion and merging.

Algorithm 1 Cluster merging via optimization direction prediction

Input: Cluster center models v, local dataset on i-th mobile client Φ_i;
Output: Merged cluster center model $\hat{v}$;

1: v_{aux}(auxiliary models), v_m(main model) $\leftarrow$ divide v;
2: $\vec{v_m} \leftarrow L1(v_{aux}, v_m)$ /* Formulate the assumption for the optimization direction as $\vec{v_m}$.; */
3: $v_m{}' \leftarrow F(v_m, \Phi_i)$; /* Train v_m on the local dataset Φ_i */
4: $\vec{v_m}' \leftarrow L1(v_m{}', v_m)$;/* Generate posterior distribution $\vec{v_m}'$ for optimization direction */
5: $\alpha \leftarrow Max\{\vec{v_m} \odot \vec{v_m}', 0\} \odot Max\{\vec{v_m} \odot \vec{v_m}'\}^{-1}$;/* Utilize $\vec{v_m}'$ and $\vec{v_m}$ to generate an aggregate attention map α. */
6: $\hat{v} \leftarrow \alpha \odot v_{aux} + (1 - \alpha) \odot v_m$ /* Employ α to aggregate v_m and v_{aux}. */

of their model parameters as the center of the merged cluster would be sub-optimal [22, 68]. To mitigate this, knowledge distillation can avoid direct manipulation of weight parameters. However, conducting the distillation process typically involves a training phase. If it is done on the client side, it will introduce training latency for slow devices and increase memory usage. Alternatively, if distillation is performed on the server side, it may compromise asynchronous efficiency due to concurrent distillation processes.

To this end, as shown in Fig. 6a, we continue to employ a training-free weight aggregation approach, instead of distillation, but we leverage the prediction of optimization directions for aggregating the cluster center model parameters to avoid a decline in accuracy. This approach ensures that there are no extra costs on the client side, and model aggregation on the server side is more efficient. In particular, we present Algorithm 1 to illustrate the key steps: In contrast to direct average aggregation, we distinguish between the main model, labeled as v_m, and the auxiliary model, denoted as v_{aux}. Empirically, we designate the model associated with a higher number of clients within the cluster as the main model v_m (Line 1). Then we extract valuable knowledge from v_{aux} and incorporate them into v_m. Here, we treat the parameters of v_{aux} as an optimization target for v_m and calculate an optimization direction $\vec{v_m}$ (Line 2). And we refine our optimization direction by incorporating the posterior distribution. Specifically, to evaluate the effectiveness of using v_{aux} as an optimization target, we first perform local training on the main model v_m using the local dataset Φ_i, expressed as $v_m{}' \leftarrow F(v_m, \Phi_i)$(Line 3). Subsequently, we calculate the weight difference between the model before training (v_m) and after training ($v_m{}'$). This process results in a posterior distribution $\vec{v_m}'$ for the optimization direction (Line 4). We leverage the weight-granularity attention matrix α in the refinement process(Line 5). With this matrix, we can carry out weight-granularity aggregation on both v_m and v_{aux}, resulting in the merged outcome $\hat{v}$ (Line 6).

4.3.3 Mobile Cluster Expansion. As shown in Fig. 6b, during the expansion process, cluster expansion extends the client with the wrong cluster into a new cluster. If the client feedback is within the same cluster, it implies the current cluster would not fit all the mobile clients. In this case, EchoPFL would split the cluster into two. The server ranks the collected client feedback in ascending order. If the client's feedback constitutes the last 20%, it will be assigned to a new cluster $\hat{c}$. The client γ assigned to the new cluster is removed from the original cluster c, while others remain in the original cluster.

Table 2. Comparison of model dissemination strategies and the communication cost of different FL paradigms.

Approachs	When ?	To whom ?	Comm optimization
FedAvg(Syn FL)	Wait for all devices	Broadcast to all	×(high cost & peak)
Oort(Syn FL)	Wait for all devices	Broadcast to all	✓(peak)
FedAsyn(Asyn FL)	Every local updates	Unicast to a device	×(high cost)
ClusterFL(Syn FL)	Wait for all devices	broadcast to a cluster	×(high cost & peak)
EchoPFL(Asyn PFL)	On-demand after every local updates	Adaptive broadcast to a cluster	✓

To rapidly learn the model parameters for the new cluster, we consider the new cluster as the original one with data drifts. Accordingly, we employ transfer learning that specifically targets domain adaptation to fine-tune the new cluster's model parameters upon those of the original cluster (line 3). However, the newly expanded cluster model obtained via transfer learning often suffers from overfitting issues [37]. This is because a scarcity of data samples from the new data drifts can lead to overfitting of the cluster center model to a small amount of new data, thereby reducing model generalization. To tackle this issue, we propose that each mobile client in a newly expanded cluster conducts local training with partial fine-tuning, focusing on adjusting the final layer output rather than full training. This partial fine-tuning restriction will only be lifted after the next cluster merging refinement, allowing the transition to normal full training mode. Technically, we assign a boolean index to label each client's local training mode within the newly expanded cluster, indicating whether they should perform partial fine-tuning or full training.

5 IN-CLUSTER MODEL BROADCAST

After client clustering, the server disseminates the latest models to clients engaged in training the same personalized model *i.e.*, within the same cluster, for staleness control. Since client clustering effectively restrains the broadcast scope and mitigates the data heterogeneity, the core of in-cluster model broadcast is how to determine the broadcast frequency to further reduce the communication cost without compromising training accuracy. First, we examine the benefits of model broadcast (Sec. 5.1) for controlling staleness, aiming to prevent accuracy deterioration and slower convergence. Subsequently, we introduce our online broadcast frequency prediction scheme in Sec. 5.2.

5.1 Advantages of In-cluster Model Broadcast

Tab. 2 compares the model dissemination strategy and the associated communication cost of different FL paradigms. Existing FL paradigms, either synchronous or asynchronous, exhibit *symmetric* client-server communication. This is not aligned with the *asymmetric bandwidth* in wireless networks, and thus *under-utilization* of the *downstream* bandwidth. In contrast, EchoPFL takes advantage of the abundant downstream bandwidth to distribute the latest models to clients when necessary, preventing accuracy decline or convergence slowdown when the staleness problem is significant. We empirically (Sec. 7.2.2) show that the asymmetric server-client communication pattern in EchoPFL not only results in decreased overall communication cost, but also avoids severe communication peaks in existing FL paradigms [28, 39, 44, 63].

As a separate note, the staleness problem, as exemplified in [53], always leads to reduced accuracy when aggregating out-of-date weights from stragglers. This is a common issue because, in practical applications, the updates from mobile local models inevitably reach the server in different epochs. Similar to [26], we represent the convergence rate for EchoPFL's asynchronous PFL by $O(\sqrt{Q_{max}Q_{avg}})$. Where, Q_{max} denotes the maximum staleness degree of models uploaded across the entire asynchronous PFL process, and Q_{avg} signifies the average of that. Aggregating a few outdated models can significantly increase Q_{max}, posing a bottleneck for convergence rate optimization. Broadcast can effectively reduce Q_{max} by distributing the cluster center model to clients, thereby improving the convergence rate.

5.2 Online In-cluster broadcast Frequency Prediction

Within each cluster, EchoPFL finetunes the broadcast frequency to further balance the communication cost and model accuracy. Rather than broadcast at fixed intervals or rounds, EchoPFL decides when to broadcast dynamically.

5.2.1 RNN-based broadcast Frequency Predictor. Our strategy, termed *model broadcast*, distributes the latest model to clients within the same cluster, proactively controlling model staleness. The broadcast scope is automatically managed by client clustering algorithms. The *broadcast frequency* is *dynamically set*. Specifically, broadcast decisions are made after each model aggregation, by comparing the *accumulated* model changes since the last broadcast and predicted model change following the next model aggregation. Broadcast is invoked when the predicted model change exceeds the accumulated model changes, *i.e.*, $L_1(\hat{v}^{(t+1)}, v^{(t)}) > L_1(v^{(t)}, v_m^{(t)})$, where $\hat{v}^{(t+1)}$, $v^{(t)}$, and $v_m^{(t)}$ are the predicted aggregated model at time $t+1$, the aggregated model at time t, and the last broadcast model till time t, respectively. The rationale is that model changes between successive aggregations diminish in convergent training [32, 39]. A substantial model change indicates intolerable model staleness. The aggregation of such models into the cluster center models has become a bottleneck in minimizing the maximum staleness Q_{max}, which is known to slow down the convergence rate. Therefore, model broadcasting becomes necessary. For simplicity, we measure the model changes in L1 distance and adopt a naive recurrent neural network to predict $\hat{v}^{(t+1)}$ based on historical models after each aggregation.

This method allows us to capture the most recent and substantial updates. K is proportional to the current number of clients within the cluster. To save storage, we keep the change degree (*e.g.*, L1-distance) of these K models rather than their model parameters.

We represent the set of these K records at each round t as $\mathbb{K}(t)$. We input these Top-K alterations sequentially into an RNN model to decide whether to broadcast for a cluster. The input length is the number of clients in the cluster. The RNN model has two hidden layers, each with 128 units. We use 1,200 historical states to pre-train the RNN model and the last K states for online fine-tuning. The training loss in RNN is:

$$
\begin{aligned}
&loss = CrossEntropy(\mathcal{P}(\mathbb{K}(t-1)), \mathcal{G}(\mathbb{K}(t))) \\
&\mathcal{G}(\mathbb{K}(t)) = \begin{cases} 1(\text{broadcast}), & \text{if } h(v_c^{t-1}, v_c^t, v_{broadcast}^{t-1}) \geq 0 \\ 0(\text{not broadcast}), & \text{if } h(v_c^{t-1}, v_c^t, v_{broadcast}^{t-1}) < 0 \end{cases} \\
&h(v_c^{t-1}, v_c^t, v_{broadcast}^{t-1}) = L1(v_c^{t-1}, v_{broadcast}^{t-1}) - L1(v_c^{t-1}, v_c^t)
\end{aligned}
\tag{4}
$$

where $\mathcal{P}(\mathbb{K}(t-1))$ is the prediction made by the RNN-based predictor on the historical records of K alterations after communication round $(t-1)$. $\mathcal{G}(\mathbb{K}(t))$ is the ground truth at $t-th$ round, which is obtained by computing the L1 distance between the weights of the previously broadcasted model $v_{broadcast}^{t-1}$ and the newly aggregated model v_c^{t-1} before $t-th$ round (*i.e.*, the accumulated gap in model staleness), as well as the L1 distance between the newly aggregated model v_c^{t-1} and the next iteration's aggregated model v_c^t (*i.e.*, the eliminated model staleness). This formulation dynamically balances broadcast frequency and model accuracy. Consequently, EchoPFL broadcasts more frequently given notable model changes; and less frequently otherwise.

5.2.2 Dynamic Predictor Maintenance. Since EchoPFL dynamically expands and merges clusters, it is crucial to continuously refine the Top-K records, RNN models, and broadcast strategies for each evolving cluster. EchoPFL maintains the necessary states for the predictors of each cluster as follows.

- *Predictor Maintenance in Cluster Expansion.*
  - The expanded cluster *resets Top-K historical records* as the newly expanded client and pads zeros since existing historical records are inapplicable to the new cluster.
  - The expanded cluster *inherits* the RNN model weights as the initial weights.

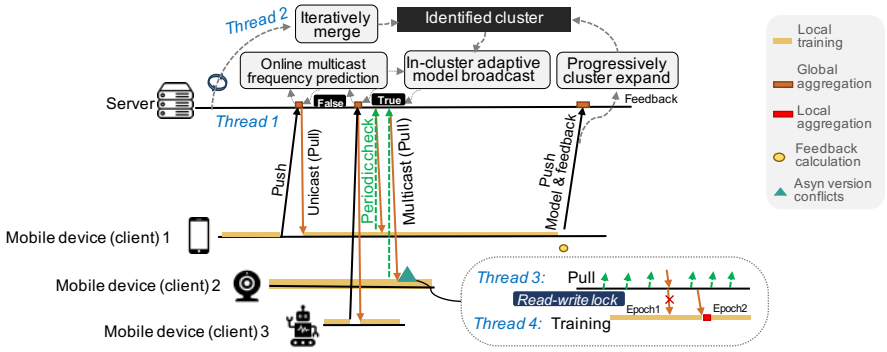


Fig. 7. Continuous integration-based client-server asynchronous coordination mechanism.

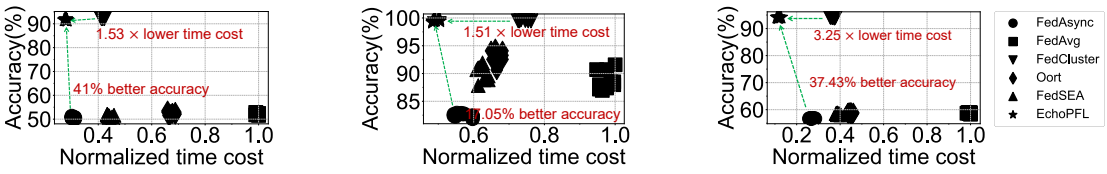
  - *broadcast* is deactivated since the cluster center weight is already up-to-date before expansion.
- *Predictor Maintenance in Cluster Merging.*
  - The merged cluster *refreshes Top-K records* by sampling distinct records from each cluster before merging. The sampling ratio is set proportional to the variance of the cached K L1 distances in each cluster. It prioritizes the selection of Top-K records with larger weight changes.
  - We adopt knowledge distillation to merge the RNN weights of multiple clusters as Sec. 4.3.2.
  - The merged cluster model is *immediately broadcast* to its clients because cluster merging induces drastic weight changes and thus model staleness.

In summary, the online predictor, coupled with the flexibility to adjust predictors and the value of K, empowers EchoPFL to adapt to heterogeneous and dynamic mobile scenarios.

6 CI-BASED VERSION CONTROL IMPLEMENTATION

In the APFL system, it is common for multiple mobile clients to update the global model simultaneously, or for new global model updates to be sent during local training, which could lead to version conflicts. We draw inspiration from the continuous integration (CI) mechanism in Git, to implement FedOM as an easy-to-use client-server coordination scheme for integration with other FL frameworks. The CI system mechanism offers specific advantages for APFL's ubiquitous applications: i) Conflict resolution: The CI mechanism adeptly handles version conflicts by controlling the aggregation of multiple model updates. ii) Immediate server feedback: Clients receive prompt feedback about their uploads, mitigating and addressing errors stemming from heterogeneities in later training stages and enhancing convergence. iii) Fast mobile application release: Mobile clients can promptly deploy services with newly updated models, minimizing the interval between global model training and its availability to mobile users. iv) Efficient branch collaboration: CI facilitates efficient collaboration in different branches, which, in the APFL context, are clusters. We present the following operations, as shown in Fig. 7:

- EchoPFL ***Pull****: Fetch models as desired.* Clients periodically query the server for significant model changes and fetch the latest model from the server to the mobile device for synchronization.
- EchoPFL ***Push****: Upload local updates on-demand.* Clients upload their local models to the server, and the server aggregates the received local models accordingly.
- EchoPFL ***Branch****: Identify personalized clusters.* Clustering is essential in personalized FL as it creates customized and targeted model updates for groups of clients with similar data characteristics. We use clustering to identify clusters of clients with similarity, and we implement these clusters' model updates as branches. We utilize multi-thread and read-write locks to resolve conflicts among personalized branches.



(a) CIFAR-10 (b) HAR (c) Ubisound

Fig. 8. Comparison of accuracy *vs.* training time between EchoPFL and other baselines on diverse tasks.

7 EXPERIMENT

7.1 Experiment Setup

Implementation. We implement EchoPFL using Python 3.7 and PyTorch 1.10 for the server and mobile clients, respectively. The server is equipped with two RTX 3080 GPUs and 128GB RAM. We use 20 mobile and embedded devices of five types: Jetson Nano (D_1), Jetson NX Xavier (D_2), Jetson Nano Orin (D_3), Jetson AGX Xavier (D_4), and Raspberry Pi 4 (D_5). They represent diverse computing capabilities and form a distributed FL system. For the implementation of EchoPFL, we set the hyperparameter C to 2 in Sec. 4.2.1. For the RNN implementation in Sec. 5.2.1, We use two hidden layers to construct the RNN model, with each layer consisting of 128 units. We pre-train the RNN model using 1, 200 historical states, and for online fine-tuning, we utilize the last K states. The K value is set to 10.

Tasks, Datasets, and Models. We experiment with four real-world mobile applications. And the data assigned to each client is Non-IID and unbalanced.

- **Image Recognition** (T_1) is ubiquitous in smart cameras/robots. We employed the CIFAR-10 dataset [27]. For Non-IID setting, each device contains 2-class data, and the data within each class can be unbalanced. The model has two convolutional (`conv`) layers followed by a fully connected (`fc`) layer.
- **Human Activity Recognition, HAR (T_2)** on mobiles has gained significant attention [71]. We adopt the HAR-UCI dataset [1], which comprises sensor data from 30 users. The model contains two `fc` layers.
- **Sound Detection (T_3)** for hard-of-hearing people using wearables is crucial. We use the Ubisound [49] dataset, comprising nine sound classes. For Non-IID setting, each device contains 3-class data, and the data within each class can be unbalanced. The model contains two `conv` layers followed by two `fc` layers.
- **Automatic Image File Cleaning** (T_4) helps users mange image files. We collected a dataset of 15000 images from phones, robots, and dashboard cameras. For Non-IID setting, devices hold unbalanced "Delete/Retain" data. The model has two `conv` layers and two `fc` layers.

Assessing the performance of the FL system can be challenging when dealing with hundreds of physical devices. Therefore, we divide our experiments into simulation experiments and real-world experiments. For simulation experiments(T_1, T_2 and T_3), we gather data on local training times and communication overhead to simulate the system using software. In detail, we simulate 20% D_1, 20% D_2, 20% D_3 and 40% D_5. For real-world experiments(T_4), we conducted them with $3D_1$, $5D_2$, $4D_3$,$2D_4$ and $6D_5$.

Baselines. We adopt six mainstream FL algorithms with mobile devices as performance comparison baselines. They are configured as follows:

- **Synchronous FL**: the server waits for all mobile clients for each round. It sets the accuracy baseline because it has the most comprehensive knowledge from all clients. It also sets a tough communication cost line due to its infrequent communication frequency.
  - **FedAvg** [39]: The server calculates the average of all mobile clients' weights. The updated global model is then broadcast back to all clients.

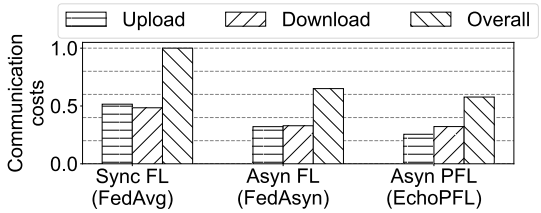
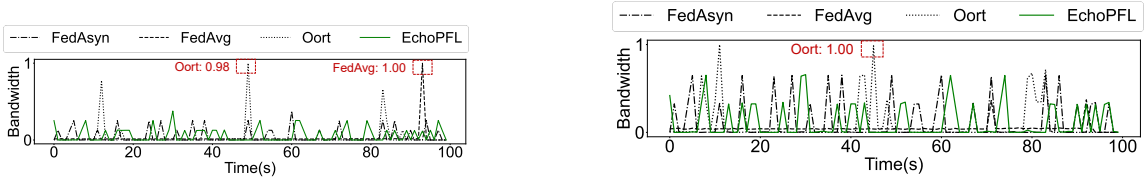


Fig. 9. EchoPFL *vs.* other baselines in terms of upload, download, and overall communication cost.



(a) Download (b) Upload

Fig. 10. Communication curve in (a) download and (b) upload processes.

– **Oort** [28] adopts mobile client selection to reduce the waiting time due to system heterogeneity.

- **Asynchronous FL: FedAsyn** [63] promptly aggregates the model and distributes updates to mobile clients in an asynchronous manner.
- **Semi-asynchronous FL: FedSEA** [53] balances model accuracy and training latency through scheduling synchronization points. It also optimizes the error caused by weight discarding of slow devices.
- **Synchronous PFL: ClusterFL** [44] trains multiple personalized models by clustering clients based on the similarity of their model outputs. And it identifies personalized clusters based on their similarities.
- **Asynchronous PFL:** EchoPFL integrates PFL into the asynchronous framework.
- **Standalone**: involves individual training on each client, without federated training with other clients.

7.2 Performance Comparison

7.2.1 Model Accuracy vs. Training Latency. We test three typical mobile tasks: image recognition (T_1), activity recognition (T_2), and sound detection (T_3). In this experiment, we use the simulation environment including 20% D_1, 20% D_2, 20% D_3 and 40% D_5. Fig. 8 shows the results. First, EchoPFL exhibits the best balance between model accuracy and training time compared to the baselines. Second, EchoPFL outperforms the four non-personalized baselines in accuracy. It achieves comparable accuracy to ClusterFL, the state-of-the-art PFL method, across all three tasks. For instance, on Ubisound, EchoPFL demonstrates accuracy improvements of 37.4%, 35.4%, 35.8%, and 35.6% compared to FedAsyn, FedAvg, Oort, and FedSEA, respectively. Third, EchoPFL consistently yields the lowest training time across all three tasks. For CIFAR-10, its training was up to 3.7× faster than the baselines. This efficiency enhancement during convergence is attributed to personalized FL's capacity to forego convergence directions irrelevant to diverse personalized models.

Summary. EchoPFL achieved the best overall trade-off between model accuracy and training time. This makes EchoPFL a promising solution for federated learning in the presence of data and system heterogeneity in ubiquitous mobile applications.

7.2.2 Communication Cost. We test the communication efficiency of EchoPFL despite its broadcast strategy.

- *Overall Communication Cost.* We compare EchoPFL's communication cost with FedAvg, FedAsyn, and FedSEA in the image recognition (T_1) task. The models are deployed on five clients: one D_1, two D_2, and

Table 3. Communication frequency and training time.

	Communication frequency (per minute upload)	**Training time** (min)
Syn FL (FedAvg)	29.22	398
Asyn FL (FedAsyn)	64	102
EchoPFL	61.8	81

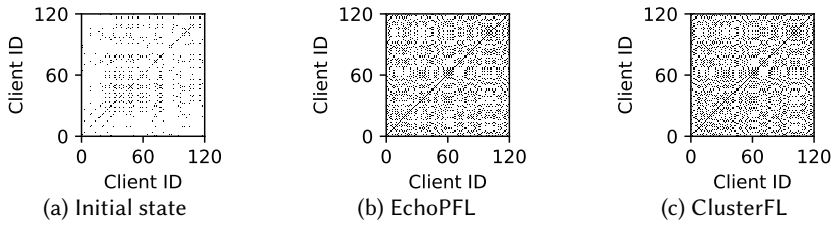


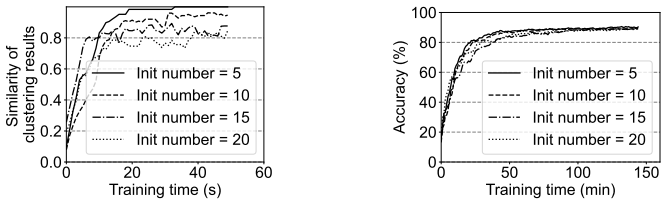
Fig. 11. Comparing EchoPFL with ClusterFL [44] in clustering results among 120 clients, visualized by a boolean matrix of collaboration relationships.

two D_4. We focus on the sum of upload and download communication costs. As depicted in Fig. 9, we normalize data by dividing by the maximum value. As a result, EchoPFL achieves a 37% reduction in overall communication costs compared to FedAvg, a 25% reduction compared to FedSEA, and comparable communication costs to FedAsyn.

- *Upload and Download Communication Cost.* To understand why EchoPFL reduces the overall communication cost, we assess the specific costs associated with upload and download communications. As shown in Fig. 10 and Tab. 3, EchoPFL reduces training time by 79.6%, significantly decreasing overall communication cost, even though it results in a 3.12× higher download frequency than FedAvg. Compared to FedAsyn, EchoPFL exhibits 1.42× more download frequency but achieves a 20.6% lower convergence time. Data in Fig. 10 is normalized by dividing by the maximum value. This indicates that increasing the download frequency does not increase the training time; instead, it brings faster convergence.
- *Communication Peak.* We compared communication peaks between EchoPFL and three baselines over a monitoring duration of 1 hour. Communication peaks can often result in packet loss and communication disruptions. As illustrated in Fig. 10, both FedAvg and Oort exhibit frequent communication peaks, mainly due to their short-term synchronous download strategies. In contrast, EchoPFL maintains relatively stable upload and download communication costs. Specifically, the upload communication peak in EchoPFL is 1.48 × lower than that of Oort and 2.08 × lower than FedAvg. This difference is attributed to EchoPFL's prevention of large-scale simultaneous model distribution, which is a common characteristic of synchronous methods like FedAvg and Oort (see Fig. 10(a)).

In summary, EchoPFL establishes a bandwidth-friendly solution for mobile FL with several advantages: *i)* EchoPFL boasts the lowest overall communication cost compared to FedAvg, FedAsyn, and FedSEA, attributed to its in-cluster broadcast method, which accelerates convergence and reduces the number of communication rounds. *ii)* While EchoPFL experiences higher download costs, it significantly curtails the upload cost, aligning well with the characteristics of asymmetrical wireless networks. *iii)* EchoPFL effectively eliminates communication peaks observed in synchronous FL.

7.2.3 Intermediate Clustering Result. We use the synchronous clustering results of ClusterFL [44] for comparison because ClusterFL can access all client weights for optimal clustering. We test with task T_1. First, as shown in Fig. 11, the clusters identified by EchoPFL are similar to those of ClusterFL. Their cosine similarity reaches up to 99% (see Fig. 12(a)), showcasing the efficacy of EchoPFL's dynamic clustering. Second, we investigate the impact of the initial cluster number in EchoPFL on the resulting clusters in Fig. 12(a). EchoPFL's is resilient to different

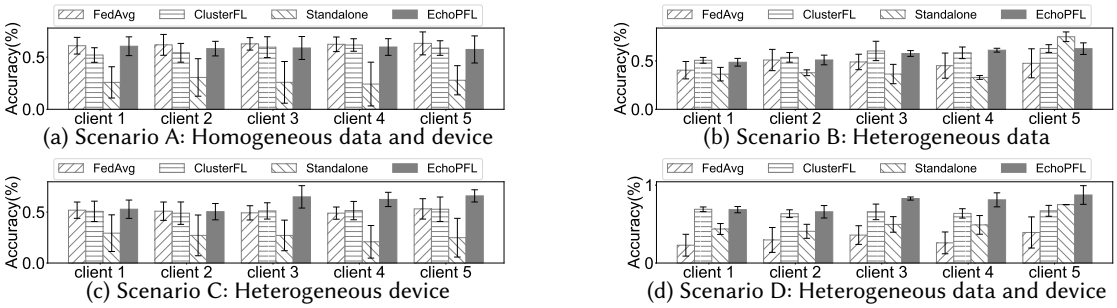


(a) Cluster similarity (b) Time to accuracy

Fig. 12. The impact of the initial number of clusters.

Table 4. Experimental data and device heterogeneity settings in four real-world scenarios.

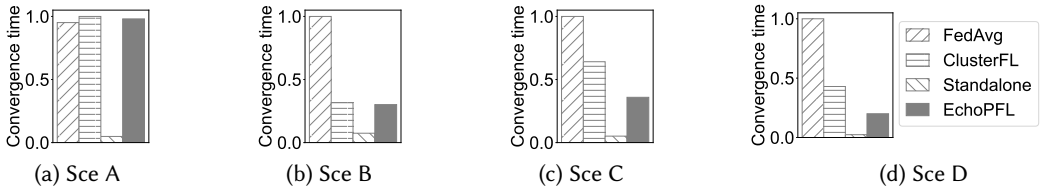
NO.	Client 1		Client2		Client3		Client4		Client5	
	Device	Data	Device	Data	Device	Data	Device	Data	Device	Data
A	D_1	10%class 1∼ 10	D_1	10%class 1∼ 10	D_1	10%class 1∼ 10	D_1	10%class 1∼ 10	D_1	10%class 1∼ 10
B	D_1	25%class1∼ 4	D_1	25%class1∼ 4	D_1	50%class1∼ 2	D_1	50%class1∼ 2	D_1	25%class1,75%class 2
C	D_1	10%class 1∼ 10	D_1	10%class 1∼ 10	D_2	10%class 1∼ 10	D_2	10%class 1∼ 10	D_4	10%class 1∼ 10
D	D_1	25%class1∼ 4	D_1	25%class1∼ 4	D_2	50%class1∼ 2	D_2	50%class1∼ 2	D_4	25%class1,75%class 2



(a) Scenario A: Homogeneous data and device (b) Scenario B: Heterogeneous data

(c) Scenario C: Heterogeneous device (d) Scenario D: Heterogeneous data and device

Fig. 13. Performance in four real-world scenarios.



(a) Sce A (b) Sce B (c) Sce C (d) Sce D

Fig. 14. Training time comparison in four scenarios.

initial cluster numbers. Furthermore, as shown in Fig. 12(b), the initial cluster number has a minimal impact (0.5%) on training time and accuracy.

7.3 Performance in Real-world Mobile Scenarios

We tested EchoPFL and three baseline methods across four real-world mobile scenarios with T_1 task. Tab. 4 outlines the configurations for data and device heterogeneity at each mobile client in the four scenarios, where diverse subsets of classes are assigned to different devices from D_1 to D_3. In Scenario B, where data heterogeneity is the focus (Fig. 13b), EchoPFL's accuracy outperforms both FedAvg and the Standalone baseline due to personalization. As data heterogeneity increases from client 1 to 5, the accuracy advantage becomes more pronounced, surpassing ClusterFL at client 5. In Scenario C, which highlights device heterogeneity (Fig. 13c), EchoPFL outperforms the baselines, mainly on fast devices (Client 3, 4, and 5). It indicates that EchoPFL enables fast devices to release highly

(a) Accuracy (b) Training time

Fig. 15. The impact of the clustering, and broadcast mechanisms on EchoPFL's performance.

Table 5. Time and accuracy differences Using L1 distance and KL divergence in real-time cluster partition

	Time for each round (s)	Trainig time (min)	Accuracy (%)
L1-distance in incremental clustering	0.0011	78.9	90.1
KL divergence in cluster adjustment	0.105		
KL divergence in incremental clustering	13.2	421.2	90.4

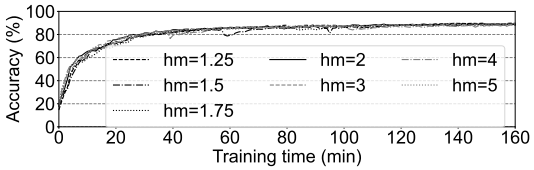


Fig. 16. Impact of *hm* in cluster merging operation.

accurate updated models promptly. In Scenario D, with both device and data heterogeneity (Fig. 13d), EchoPFL achieves notably higher accuracy than the baselines. It is attributed to EchoPFL's personalization capability and its asynchronous mechanism. Furthermore, across all these scenarios, EchoPFL consistently exhibits the shortest training time, as demonstrated in Fig. 14. The convergence time is normalized by dividing by the maximum value.

7.4 Ablation and Micro-benchmark

This subsection validates EchoPFL's module and explores different hyperparameter settings. In this experiment, we use T_1 task with simulation experiment setting(20% D_1, 20% D_2, 20% D_3 and 40% D_5).

7.4.1 w/ vs. w/o clustering: we assess the necessity and impact of the dynamic client clustering method in EchoPFL, which is responsible for customizing personalized models. As depicted in Fig. 15, without the dynamic clustering block, the accuracy is reduced to be similar to FedAvg.

7.4.2 w/ vs. w/o broadcast. We validate the significance of the in-cluster model broadcast method in EchoPFL. As shown in Fig. 15, without the broadcast method, the accuracy decreased by 8.09%, and the training time is prolonged by 1.8×.

7.4.3 Distance measure choice in real-time cluster partition. We validate the choice of distance measures in client clustering. As shown in Tab. 5, using L1 distance results in decreases up to 5× training time compared to KL divergence, satisfying the real-time demands. And leveraging KL divergence for the merge step allows EchoPFL to achieve high final accuracy.

7.4.4 Cluster number hyperparameter in merging. As discussed in Sec. 4.3, the maximized cluster number hyperparameter plays a crucial role in determining when should trigger the merge operation. This experiment validates EchoPFL's robustness to diverse hyperparameters *hm*× initial cluster number for triggering the merge

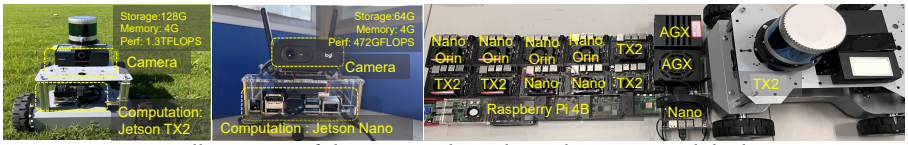


Fig. 17. Illustration of the case study with 20 ubiquitous mobile devices.

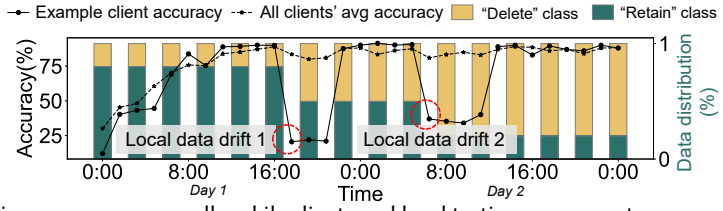


Fig. 18. Average local testing accuracy across all mobile clients and local testing accuracy at an example client under Non-IID data with dynamic distribution shifts.

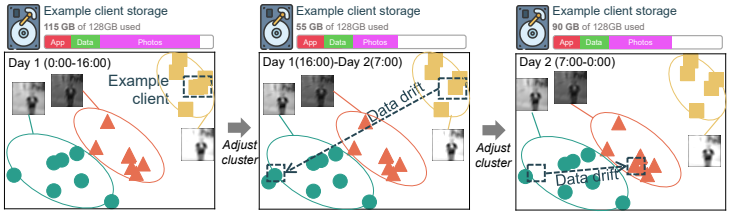


Fig. 19. Visualizing dynamically adjusted clusters.

operation. Fig. 16 demonstrates that EchoPFL is robust and insensitive to the value of *hm* in convergence time and accuracy. Thus we set *hm* to 2 by default in EchoPFL.

7.5 Real-world Case Study

We adopt 20 mobile and embedded devices(D_1, $5D_2$, $4D_3$,$2D_4$ and $6D_5$) to conduct a two-day study with the automatic image file cleaning applications (T_4), as shown in Fig. 17. This app can help users automatically clean redundant images/videos in the image file library. We employed 20 participants to label each image sample collected locally from 20 embedded devices as either "Delete" or "Retain" class based on their preferences, which resulted in Non-IID data distributions across devices. We gather data through random snapshots taken by autonomous vehicles. This data is divided into 20 segments and labeled by 20 users. The labeled data from each device is split into testing and training data by 2:8. In addition, to validate the adaptability of EchoPFL, we directed participants to change their labeling preferences for the "retain/delete" classes twice during the 2-day study, thereby simulating local data distribution shifts.

Fig. 18 shows an example of client's data distribution shift on Day 1 (16:00) and Day 2 (7:00). In this two-day study, the example client's models dynamically adapted to the changing data distribution during federated learning. The average local testing accuracy across all Non-IID clients remained stable and consistently exceeded 80%, affirming EchoPFL's efficacy in managing diverse data. Furthermore, during the shifts in local data distribution, the accuracy notably dropped at 16:00 on Day 1 and 7:00 on Day 2. Our approach promptly responded with adjustments over 2-3 rounds of federated training, allowing the personalized models to adapt to the new data. Consequently, the local testing accuracy swiftly rebounded to 89.3%. We also showcased the adaptability in EchoPFL's clusters in Fig. 19. To visualize, we adopt principal component analysis to reveal how EchoPFL's clusters adapt to the changing data distribution of this example client.

8 RELATED WORK

FL in Ubiquitous Applications. Recently, federated learning (FL) [39] has been widely applied in mobile scenarios [46]. Researchers deployed FL in various mobile scenarios, including transportation [20, 35, 57], recommendation systems [42], activity recognition [43, 44] and robotics [36]. For example, Niu *et al.* [42] employed FL to enhance the recommendation system within a mobile system operating at a billion-scale. To ease development across diverse mobile applications, Beutel *et al.* [2] proposed a system framework called Flower.

Personalized FL. Due to Non-IID and unbalanced client data, personalized FL (PFL) is proposed [54] to output multiple personalized models instead of a single one. Specifically, PFL contains two main categories: global model personalization and learning personalized models. The former always involves "FL training + local adaptation" steps. It mainly contains local fine-tuning [59, 66, 68] and meta-learning [16]. For example, Wang *et al.* [59] propose to fine-tune the head layers of the global model to realize personalization. The latter introduces methods like clustering [4, 17, 44], multi-task learning [50], and model interpolation [19]. Additionally, [23] introduced dynamic clustering to achieve personalization in the presence of various types of drift. EchoPFL focus on clustering-based PFL algorithms arises from the high clusterability observed in the data distributions across many mobile applications [6, 44, 51]. Also, the cluster-based methods for handling device heterogeneity can be applied to handle these drifts as well.

Synchronous and Asynchronous FL. Device heterogeneity refers to the diversity in computational resources. It can be addressed by modifying the model architecture and system-level adaptation. The former assigns lightweight models to devices with lower resources [13, 21, 31, 47, 55]. For example, Li *et al.* [31] utilize knowledge distillation to aggregate models. The latter mainly fall into client selection-based methods [28, 30], asynchronous FL (AFL)[9, 18, 63], and semi-asynchronous FL (SAFL) methods [53, 61]. PyramidFL [30] selects clients with similar performance to participate in the same training round. [62] present to adjust the broadcasting timing before or after local training in a synchronous setting, with a fixed broadcast frequency. For SAFL, Sun *et al.* [53] propose FedSEA, establishing periodic synchronization points to mitigate the significant impact caused by stragglers. For AFL, Xie *et al.* [63] let the server immediately aggregate uploaded weights from each client while compromising the knowledge aggregation from slow devices. EchoPFL embraces the asynchronous paradigm for its latency benefits and also addresses its shortcoming in aggregating data from stragglers by employing on-demand broadcast.

9 CONCLUSION

This paper presents EchoPFL, a client-server coordination mechanism for asynchronous personalized FL (PFL) via on-demand model broadcast. EchoPFL is the first work that effectively integrates the asynchronous mechanism into PFL, ensuring the inclusion of all fast or slow mobile clients without sacrificing accuracy. It incrementally creates and manages clusters based on the incoming model updates and feedback. And it predicts the optimal broadcast frequency to further reduce the downstream communication cost without compromising accuracy. Evaluations on four popular mobile tasks and real-world scenarios over twenty mobile devices show that EchoPFL achieves training time reductions of up to 88.2%, accuracy improvements of up to 41.04%, and communication cost reduction of up to 37%. EchoPFL enables diverse ubiquitous devices to efficiently participate in the federated learning process, meeting personalization needs, and providing systematic support for the universal application of Federated Learning (FL) systems. In future work, we aim to improve EchoPFL's compatibility with various PFL algorithms, addressing feature and concept skew. Additionally, we intend to integrate EchoPFL into deep learning frameworks, facilitating the smooth deployment of asynchronous PFL in real-world ubiquitous applications.

ACKNOWLEDGMENTS

This work was partially supported by the National Science Fund for Distinguished Young Scholars (62025205) the National Natural Science Foundation of China (No. 62032020, 62102317), and CityU APRC grant No. 9610633. The authors thank Lei Wu for the mathematical discussions about EchoPFL and the anonymous reviewers for their constructive feedback that has made the work stronger.

REFERENCES

[1] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, Jorge Luis Reyes-Ortiz, et al. 2013. A public domain dataset for human activity recognition using smartphones.. In *Esann*, Vol. 3. 3.

[2] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Kwing Hei Li, Titouan Parcollet, Pedro Porto Buarque de Gusmão, et al. 2020. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390* (2020).

[3] Siddharth Bhatia, Bryan Hooi, Minji Yoon, Kijung Shin, and Christos Faloutsos. 2020. Midas: Microcluster-based detector of anomalies in edge streams. In *Proceedings of AAAI*, Vol. 34. 3242–3249.

[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 *Proceedings of IJCNN*. IEEE, 1–9.

[5] Feng Cao, Martin Estert, Weining Qian, and Aoying Zhou. 2006. Density-based clustering over an evolving data stream with noise. In *Proceedings of SDM*. SIAM, 328–339.

[6] Liang Cao, Yufeng Wang, Bo Zhang, Qun Jin, and Athanasios V Vasilakos. 2018. GCHAR: An efficient Group-based Context—Aware human activity recognition on smartphone. *J. Parallel and Distrib. Comput.* 118 (2018), 67–80.

[7] Daoyuan Chen, Dawei Gao, Weirui Kuang, Yaliang Li, and Bolin Ding. 2022. pFL-bench: A comprehensive benchmark for personalized federated learning. *Advances in Neural Information Processing Systems* 35 (2022), 9344–9360.

[8] Shanzhi Chen and Jian Zhao. 2014. The requirements, challenges, and technologies for 5G of terrestrial mobile telecommunication. *IEEE communications magazine* 52, 5 (2014), 36–43.

[9] Yujing Chen, Yue Ning, Martin Slawski, and Huzefa Rangwala. 2020. Asynchronous online federated learning for edge devices with non-iid data. In *Proceedings of Big Data*. IEEE, 15–24.

[10] Hyunsung Cho, Akhil Mathur, and Fahim Kawsar. 2022. Flame: Federated learning across multi-device environments. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 3 (2022), 1–29.

[11] Sebastian Claici, Mikhail Yurochkin, Soumya Ghosh, and Justin Solomon. 2020. Model fusion with Kullback-Leibler divergence. In *International conference on machine learning*. PMLR, 2038–2047.

[12] Yongheng Deng, Weining Chen, Ju Ren, Feng Lyu, Yang Liu, Yunxin Liu, and Yaoxue Zhang. 2022. TailorFL: Dual-Personalized Federated Learning under System and Data Heterogeneity. In *Proceedings of Sensys*. 592–606.

[13] Enmao Diao, Jie Ding, and Vahid Tarokh. 2020. Heterofl: Computation and communication efficient federated learning for heterogeneous clients. *arXiv preprint arXiv:2010.01264* (2020).

[14] Youngwook Do, Jung Wook Park, Yuxi Wu, Avinandan Basu, Dingtian Zhang, Gregory D Abowd, and Sauvik Das. 2021. Smart webcam cover: exploring the design of an intelligent webcam cover to improve usability and trust. *Proceedings of IMWUT* 5, 4 (2021), 1–21.

[15] Chen Dun, Mirian Hipolito, Chris Jermaine, Dimitrios Dimitriadis, and Anastasios Kyrillidis. 2023. Efficient and Light-Weight Federated Learning via Asynchronous Distributed Dropout. In *Proceedings of AISTATS*. PMLR, 6630–6660.

[16] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. 2020. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. *Advances in Neural Information Processing Systems* 33 (2020), 3557–3568.

[17] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. 2020. An efficient framework for clustered federated learning. *Advances in Neural Information Processing Systems* 33 (2020), 19586–19597.

[18] Bin Gu, An Xu, Zhouyuan Huo, Cheng Deng, and Heng Huang. 2021. Privacy-preserving asynchronous vertical federated learning algorithms for multiparty collaborative learning. *IEEE transactions on neural networks and learning systems* 33, 11 (2021), 6103–6115.

[19] Filip Hanzely and Peter Richtárik. 2020. Federated learning of a mixture of global and local models. *arXiv preprint arXiv:2002.05516* (2020).

[20] Yuze He, Li Ma, Jiahe Cui, Zhenyu Yan, Guoliang Xing, Sen Wang, Qintao Hu, and Chen Pan. 2022. AutoMatch: Leveraging Traffic Camera to Improve Perception and Localization of Autonomous Vehicles. In *Proceedings of Sensys*. 16–30.

[21] Samuel Horvath, Stefanos Laskaridis, Mario Almeida, Ilias Leontiadis, Stylianos Venieris, and Nicholas Lane. 2021. Fjord: Fair and accurate federated learning under heterogeneous targets with ordered dropout. *Advances in Neural Information Processing Systems* 34 (2021), 12876–12889.

[22] Hai Jin, Dongshan Bai, Dezhong Yao, Yutong Dai, Lin Gu, Chen Yu, and Lichao Sun. 2022. Personalized edge intelligence via federated self-knowledge distillation. *IEEE Transactions on Parallel and Distributed Systems* 34, 2 (2022), 567–580.

[23] Ellango Jothimurugesan, Kevin Hsieh, Jianyu Wang, Gauri Joshi, and Phillip B Gibbons. 2023. Federated learning under distributed concept drift. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 5834–5853.

[24] Woosub Jung, Kenneth Koltermann, Noah Helm, GinaMari Blackwell, Ingrid Pretzer-Aboff, Leslie Cloud, and Gang Zhou. 2022. IMU Sensing Data-Based Kinetic Tremor Detection in Parkinson's Disease Patients. In *Proceedings of Sensys*. 772–773.

[25] Minchul Kim, Anil K Jain, and Xiaoming Liu. 2022. Adaface: Quality adaptive margin for face recognition. In *Proceedings of CVPR*. 18750–18759.

[26] Anastasiia Koloskova, Sebastian U Stich, and Martin Jaggi. 2022. Sharper convergence guarantees for asynchronous sgd for distributed and federated learning. *Advances in Neural Information Processing Systems* 35 (2022), 17202–17215.
[27] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).
[28] Fan Lai, Xiangfeng Zhu, Harsha V Madhyastha, and Mosharaf Chowdhury. 2021. Oort: Efficient Federated Learning via Guided Participant Selection.. In *Proceedings of OSDI*. 19–35.
[29] Ang Li, Jingwei Sun, Pengcheng Li, Yu Pu, Hai Li, and Yiran Chen. 2021. Hermes: an efficient federated learning framework for heterogeneous mobile clients. In *Proceedings of MobiCom*. 420–437.
[30] Chenning Li, Xiao Zeng, Mi Zhang, and Zhichao Cao. 2022. PyramidFL: A fine-grained client selection framework for efficient federated learning. In *Proceedings of MobiCom*. 158–171.
[31] Daliang Li and Junpu Wang. 2019. Fedmd: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581* (2019).
[32] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. 2019. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189* (2019).
[33] Youpeng Li, Xuyu Wang, and Lingling An. 2023. Hierarchical Clustering-based Personalized Federated Learning for Robust and Fair Human Activity Recognition. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 7, 1 (2023), 1–38.
[34] Feng Liang, Weike Pan, and Zhong Ming. 2021. Fedrec++: Lossless federated recommendation with explicit feedback. In *Proceedings of AAAI*, Vol. 35. 4224–4231.
[35] Xinle Liang, Yang Liu, Tianjian Chen, Ming Liu, and Qiang Yang. 2022. Federated transfer reinforcement learning for autonomous driving. In *Federated and Transfer Learning*. Springer, 357–371.
[36] Boyi Liu, Lujia Wang, and Ming Liu. 2019. Lifelong federated reinforcement learning: a learning architecture for navigation in cloud robotic systems. *IEEE Robotics and Automation Letters* 4, 4 (2019), 4555–4562.
[37] Ziquan Liu, Yi Xu, Yuanhong Xu, Qi Qian, Hao Li, Xiangyang Ji, Antoni Chan, and Rong Jin. 2022. Improved fine-tuning by better leveraging pre-training data. *Advances in Neural Information Processing Systems* 35 (2022), 32568–32581.
[38] Qianpiao Ma, Yang Xu, Hongli Xu, Zhida Jiang, Liusheng Huang, and He Huang. 2021. FedSA: A semi-asynchronous federated learning mechanism in heterogeneous edge computing. *IEEE Journal on Selected Areas in Communications* 39, 12 (2021), 3654–3672.
[39] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 1273–1282.
[40] Anh Nguyen, Tuong Do, Minh Tran, Binh X Nguyen, Chien Duong, Tu Phan, Erman Tjiputra, and Quang D Tran. 2022. Deep federated learning for autonomous driving. In *Proceedings of IV*. IEEE, 1824–1830.
[41] Takayuki Nishio and Ryo Yonetani. 2019. Client selection for federated learning with heterogeneous resources in mobile edge. In *Proceedings of ICC*. IEEE, 1–7.
[42] Chaoyue Niu, Fan Wu, Shaojie Tang, Lifeng Hua, Rongfei Jia, Chengfei Lv, Zhihua Wu, and Guihai Chen. 2020. Billion-scale federated learning on mobile clients: A submodel design with tunable privacy. In *Proceedings of MobiCom*. 1–14.
[43] Xiaomin Ouyang, Zhiyuan Xie, Heming Fu, Sitong Cheng, Li Pan, Neiwen Ling, Guoliang Xing, Jiayu Zhou, and Jianwei Huang. 2023. Harmony: Heterogeneous Multi-Modal Federated Learning through Disentangled Model Training. In *Proceedings of MobiSys*. 530–543.
[44] Xiaomin Ouyang, Zhiyuan Xie, Jiayu Zhou, Jianwei Huang, and Guoliang Xing. 2021. Clusterfl: a similarity-aware federated learning system for human activity recognition. In *Proceedings of MobiSys*. 54–66.
[45] Jungwuk Park, Dong-Jun Han, Minseok Choi, and Jaekyun Moon. 2021. Sageflow: Robust federated learning against both stragglers and adversaries. *Advances in neural information processing systems* 34 (2021), 840–851.
[46] Kilian Pfeiffer, Martin Rapp, Ramin Khalili, and Jörg Henkel. 2023. Federated Learning for Computationally-Constrained Heterogeneous Devices: A Survey. *Comput. Surveys* (2023).
[47] Martin Rapp, Ramin Khalili, Kilian Pfeiffer, and Jörg Henkel. 2022. Distreal: Distributed resource-aware learning in heterogeneous systems. In *Proceedings of AAAI*, Vol. 36. 8062–8071.
[48] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. 2020. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE Transactions on neural networks and learning systems* 32, 8 (2020), 3710–3722.
[49] Liu Sicong, Zhou Zimu, Du Junzhao, Shangguan Longfei, Jun Han, and Xin Wang. 2017. UbiEar: Bringing location-independent sound awareness to the hard-of-hearing people with smartphones. *Proceedings of IMWUT* 1, 2 (2017), 1–21.
[50] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. 2017. Federated multi-task learning. *Advances in neural information processing systems* 30 (2017).
[51] Allan Stisen, Henrik Blunck, Sourav Bhattacharya, Thor Siiger Prentow, Mikkel Baun Kjærgaard, Anind Dey, Tobias Sonne, and Mads Møller Jensen. 2015. Smart devices are different: Assessing and mitigating mobile sensing heterogeneities for activity recognition. In *Proceedings of SenSys*. 127–140.
[52] Marcin Straczkiewicz, Peter James, and Jukka-Pekka Onnela. 2021. A systematic review of smartphone-based human activity recognition methods for health research. *NPJ Digital Medicine* 4, 1 (2021), 148.

[53] Jingwei Sun, Ang Li, Lin Duan, Samiul Alam, Xuliang Deng, Xin Guo, Haiming Wang, Maria Gorlatova, Mi Zhang, Hai Li, et al. 2022. FedSEA: A Semi-Asynchronous Federated Learning Framework for Extremely Heterogeneous Devices. (2022).
[54] Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. 2022. Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems* (2022).
[55] Yue Tan, Guodong Long, Lu Liu, Tianyi Zhou, Qinghua Lu, Jing Jiang, and Chengqi Zhang. 2022. Fedproto: Federated prototype learning across heterogeneous clients. In *Proceedings of AAAI*, Vol. 36. 8432–8440.
[56] Qinyong Wang, Hongzhi Yin, Tong Chen, Junliang Yu, Alexander Zhou, and Xiangliang Zhang. 2021. Fast-adapting and privacy-preserving federated recommender system. *The VLDB Journal* (2021), 1–20.
[57] Yansheng Wang, Yongxin Tong, Zimu Zhou, Ziyao Ren, Yi Xu, Guobin Wu, and Weifeng Lv. 2022. Fed-LTD: Towards cross-platform ride hailing via federated learning to dispatch. In *Proceedings of KDD*. 4079–4089.
[58] Yansheng Wang, Yongxin Tong, Zimu Zhou, Ruisheng Zhang, Sinno Jialin Pan, Lixin Fan, and Qiang Yang. 2023. Distribution-Regularized Federated Learning on Non-IID Data. In *Proceedings of ICDE*. 2113–2125.
[59] Yansong Wang, Hui Xu, Waqar Ali, Miaobo Li, Xiangmin Zhou, and Jie Shao. 2023. Fedftha: a fine-tuning and head aggregation method in federated learning. *IEEE Internet of Things Journal* (2023).
[60] Hao Wu, Jinghao Feng, Xuejin Tian, Edward Sun, Yunxin Liu, Bo Dong, Fengyuan Xu, and Sheng Zhong. 2020. EMO: Real-time emotion recognition from single-eye images for resource-constrained eyewear devices. In *Proceedings of Mobisys*. 448–461.
[61] Wentai Wu, Ligang He, Weiwei Lin, Rui Mao, Carsten Maple, and Stephen Jarvis. 2020. SAFA: A semi-asynchronous protocol for fast federated learning with low overhead. *IEEE Trans. Comput.* 70, 5 (2020), 655–668.
[62] Ming Xiang, Stratis Ioannidis, Edmund Yeh, Carlee Joe-Wong, and Lili Su. 2023. Towards Bias Correction of FedAvg over Nonuniform and Time-Varying Communications. *arXiv preprint arXiv:2306.00280* (2023).
[63] Cong Xie, Sanmi Koyejo, and Indranil Gupta. 2019. Asynchronous federated optimization. *arXiv preprint arXiv:1903.03934* (2019).
[64] Chunmei Xu, Shengheng Liu, Zhaohui Yang, Yongming Huang, and Kai-Kit Wong. 2021. Learning rate optimization for federated learning exploiting over-the-air computation. *IEEE Journal on Selected Areas in Communications* 39, 12 (2021), 3742–3756.
[65] Haibo Yang, Xin Zhang, Prashant Khanduri, and Jia Liu. 2022. Anarchic federated learning. In *International Conference on Machine Learning*. PMLR, 25331–25363.
[66] Ling-Li Zeng, Zhipeng Fan, Jianpo Su, Min Gan, Limin Peng, Hui Shen, and Dewen Hu. 2022. Gradient matching federated domain adaptation for brain image classification. *IEEE Transactions on Neural Networks and Learning Systems* (2022).
[67] Jie Zhang, Song Guo, Xiaosong Ma, Haozhao Wang, Wenchao Xu, and Feijie Wu. 2021. Parameterized knowledge transfer for personalized federated learning. *Advances in Neural Information Processing Systems* 34 (2021), 10092–10104.
[68] Lin Zhang, Li Shen, Liang Ding, Dacheng Tao, and Ling-Yu Duan. 2022. Fine-tuning global model via data-free knowledge distillation for non-iid federated learning. In *Proceedings of CVPR*. 10174–10183.
[69] Shujie Zhang, Tianyue Zheng, Hongbo Wang, Zhe Chen, and Jun Luo. 2022. Quantifying the Physical Separability of RF-Based Multi-Person Respiration Monitoring via SINR. In *Proceedings of Sensys*. 47–60.
[70] Tuo Zhang, Lei Gao, Chaoyang He, Mi Zhang, Bhaskar Krishnamachari, and A Salman Avestimehr. 2022. Federated learning for the internet of things: Applications, challenges, and opportunities. *IEEE Internet of Things Magazine* 5, 1 (2022), 24–29.
[71] Tianfang Zhang, Cong Shi, Payton Walker, Zhengkun Ye, Yan Wang, Nitesh Saxena, and Yingying Chen. 2023. Passive Vital Sign Monitoring via Facial Vibrations Leveraging AR/VR Headsets. In *Proceedings of MobiSys*. 96–109.
[72] Wenhao Zhang, Zimu Zhou, Yansheng Wang, and Yongxin Tong. 2023. DM-PFL: Hitchhiking Generic Federated Learning for Efficient Shift-Robust Personalization. In *Proceedings of KDD*. 3396–3408.
[73] Zhendong Zhuang and Yang Xue. 2019. Sport-related human activity detection and recognition using a smartwatch. *Sensors* 19, 22 (2019), 5001.