ClassTer: Mobile Shift-Robust Personalized Federated Learning via Class-Wise Clustering

Xiaochen Li, Sicong Liu*, *Member, IEEE,* Zimu Zhou, *Member, IEEE,* Yuan Xu,
Bin Guo, *Senior Member, IEEE,* Zhiwen Yu, *Senior Member, IEEE,*

Abstract—The rise of mobile devices with abundant sensor data and computing power has driven the trend of federated learning (FL) on them. Personalized FL (PFL) aims to train tailored models for each device, addressing data heterogeneity from diverse user behaviors and preferences. However, due to dynamic mobile environments, PFL faces challenges in *test-time data shifts*, *i.e.*, variations between training and testing. While this issue is well studied in generic deep learning through model generalization or adaptation, this issue remains less explored in PFL, where models often overfit local data. To address this, we introduce ClassTer, a shift-robust PFL framework. We observe that class-wise clustering of clients in cluster-based PFL (CFL) can avoid class-specific biases by decoupling the training of classes. Thus, we propose a paradigm shift from traditional client-wise clustering to *class-wise clustering*, which allows *effective aggregation* of cluster models into a generalized one via knowledge distillation. Additionally, we extend ClassTer to *asynchronous* mobile clients to optimize wall clock time by leveraging critical learning periods and both intra- and inter-device scheduling. Experiments show that compared to status quo approaches, ClassTer achieves a reduction of up to 91% in convergence time, and an improvement of up to 50.45% in accuracy.

Index Terms—Personalized federated learning, shift-robust, asynchronous mobile devices

1 Introduction

The rapid increase in sensory data from mobile devices, combined with their local computing power and widespread wireless networks, has catalyzed the emergence of federated learning (FL) on these devices. The proliferation of rich sensors on widely used mobile devices, combined with their computational capabilities and the expansive reach of wireless networks, has propelled the development of federated learning (FL) on them [2]. In FL, multiple mobile devices ranging from smartphones to drones and robots work together to train a shared deep model under centralized server management while ensuring data privacy by retaining data locally [2], [40], [90], [94]. This method is particularly advantageous for resource-intensive deep learning tasks on mobile devices [89], [95], including activity recognition (*e.g.*, Google keyboard [77]), personalized recommendations (*e.g.*, Alibaba shopping recommendation [78]), and communication optimization (*e.g.*, Meta video call app [79]). A unique feature of FL on mobile devices is its *non-IID data* resulting from diverse user behaviors and preferences, exemplified by Siemens factories employing FL in defect detection: air conditioning production lines primarily detect cracks and scratches, whereas razor lines typically encounter discoloration and deformation.

*Corresponding Author: Sicong Liu (e-mail:scliu@nwpu.edu.cn)

To address *data heterogeneity* across mobile devices, personalized federated learning (PFL) tailors models to specific device data profiles. Existing methods for PFL can be divided into two types based on the presence of a global model. without a global model, PFL methods directly train *multiple* personalized models, including clustering [12], [16], [17], [53], multi-task learning [21], [22]. With a global model, PFL methods follow global model personalization, including local fine-tuning [43], [73] or meta-learning [14], [15], [18], and model interpolation [36], [37], [92].

Yet, orthogonal to data heterogeneity across clients, data heterogeneity within a single mobile client can also differ between training and testing *over time* due to dynamic and diverse mobile environments, referred to as ***test-time data shift*** (we defer more examples in Sec. 2.2). While this issue is well-examined in generic deep learning contexts [46], [80], [81], [89], [91], such as by improving model generalization [7], [36], [47], [81] and realizing adaptation [16], [29], [46], [80], it remains less-explored in PFL, where models often overfit to local personalized training data.

In PFL, we observe that enhancing model **generalization** is favorable over *adaptation* methods for combating mobile data shifts because individual mobile clients often possess *limited sensory data*, which may be insufficient to fine-tune the model (*i.e.*, the reason for federated learning). Accordingly, previous shift-robust PFL methods [28], [39] leverage the *global model*, which is often an *intermediate result* to train the personalized models, to improve the generalization of the personalized models. However, prior research fails to achieve shift-robust PFL under *extreme data heterogeneity* in mobile application scenarios for the following reasons.

- Existing shift-robust PFL schemes [28], [65] implicitly assume the feasibility to train a global model, which may not converge with extreme non-IID mobile data [12], [56] (see more detials in Sec. 2.2).
- Previous PFL strategies, such as clustering-based FL (CFL) [17], [53], that deal with *extreme data heterogeneity* typically assume a fixed data distribution between train- and test-times. This neglect of data mismatch over time makes them vulnerable to non-stationary test-time mobile data shifts.

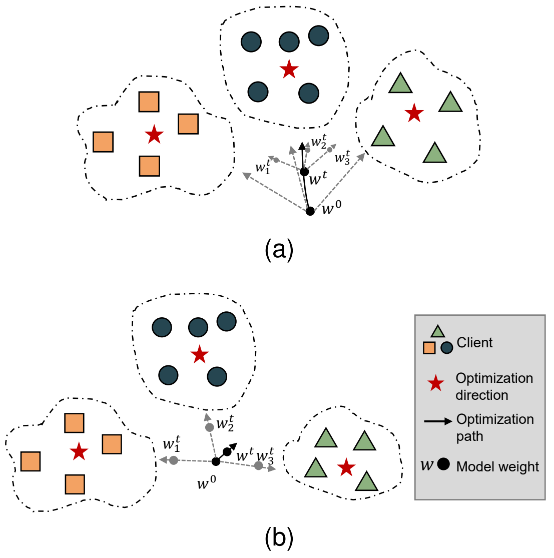


Fig. 1. Extracting a (global) generalized model under (a) mild and (b) extreme data heterogeneity in CFL. It can be challenging to aggregate the cluster models into a generalized one as in other shift-robust PFL methods, particularly under extreme data heterogeneity.

An intuitive solution is to extract a *generalized model* from PFL strategies designed for extreme data heterogeneity, *e.g.*, clustering-based personalized federated learning (CFL) [12], which groups clients with similar data distributions into clusters and learns distinct models per cluster (known as *cluster models*). It manages data heterogeneity through clustering and is prevalent in mobile applications, where mobile data often exhibit natural clusterability due to similarities in mobile users' lifestyles, work habits, and physical environments [17]. Nevertheless, extracting a generalized model during CFL is challenging. *Firstly*, there is usually no global model in CFL. More importantly, naively aggregating the cluster models into a (global) generalized model can be ineffective under extreme mobile data heterogeneity. For example, a badminton player's watch struggles to collect basketball activity data. As depicted in Fig. 1, although a generalized model may be aggregated under mild data heterogeneity (see Fig. 1a), it can be difficult, if not impossible, to extract the generalized model under extreme data heterogeneity (see Fig. 1b).

In response, we present ClassTer, a mobile data shift-robust personalized federated learning via class-wise clustering. **First**, we observe that the class-wise clustering of mobile clients in cluster-based personalized FL (CFL) can avoid class-specific biases by decoupling the training of classes (see Sec. 3.1). Therefore, we propose a paradigm shift from traditional *client-wise* clustering to *class-wise* clustering, which allows effective aggregation of the cluster models into a generalized one via knowledge distillation. Specifically, ClassTer fairly schedules clients to *contrastively* train multiple single-class models, and the server extracts a *generalized* model from these single-class models via knowledge distillation to achieve shift-robust PFL with synchronous clients. **Second**, we extend ClassTer to *asynchronous* mobile clients for faster convergence, measured by wall clock time, where clusters aggregate updates as soon as they are available from each mobile client. Supporting asynchronous mobile clients makes ClassTer more practical, as mobile clients in real-world FL systems often experience *heterogeneous* resource availability and *dynamic* network bandwidth, leading to asynchronous local model updates and staleness issue [50]. In particular, we harness the notion of *critical learning periods* (Sec. 5.1) and perform both intra- and inter-cluster scheduling to control staleness and optimize training convergence in asynchronous shift-robust PFL.

We evaluate the performance of ClassTer on 4 mobile tasks and 15 real-world scenarios with diverse data or system heterogeneity using 20 mobile devices. Results show a reduction in training time of up to 88.2% and an improvement in accuracy of up to 50.45% (Sec. 6.2) Our main contributions are summarized as follows.

- To the best of our knowledge, this is the first work that compacts data shifts with extreme Non-IID data distribution in PFL. It maintains high accuracy under extreme Non-IID data and avoids significant accuracy decline with test-time mobile data shifts.
- We propose ClassTer, shift-robust PFL via *class-wise* clustering. It adapts to extreme Non-IID data shifts using class-wise clustering and efficiently extracts a generalized global model through intra-cluster scheduling. We also harness the critical learning period to implement an asynchronous version using intra- and inter-cluster schedules to control staleness.
- Experiments show that ClassTer outperforms existing a-/synchronous or personalized FL methods [2], [28], [40], [53], [73] in trading off training accuracy and latency across various mobile tasks, data shifts, platforms, and scenarios.

2 Problem Statement

2.1 Preliminaries

Federated learning (FL) collaboratively trains a powerful deep model with the distributed data possessed by individual clients (mobile devices), which would be otherwise impossible with the limited data on a single client alone [2]. The standard FL involves multiple clients and a server. Clients train *local models* with their own data and push them to the server. The server aggregates the local models into a *global model*, which is then sent back to the clients. The iteration continues until model convergence.

Mobile Data Heterogeneity and Model Personalization. In mobile applications, attributed to diverse user behaviors and preferences, sensory data collected by each client may notably vary, leading to severe *data heterogeneity*, which can significantly impair model convergence [7], [57]. Furthermore, severe data heterogeneity implies that a single global model may not fit all the clients, necessitating personalized federated learning (PFL), which learns customized models for individual clients [55], [56]. Specifically, given clients $\{1, 2, \ldots, C\}$ holding local training datasets $\{D_1, D_2, \ldots, D_C\}$, PFL trains personalized models $\{w_1, w_2, \ldots, w_C\}$ where w_c is deployed to client c, with the following objective:

$$\min_{w_1, w_2, \ldots, w_C} l(w_1, \ldots, w_C) = \sum_{c=1}^{C} \frac{|D_c|}{|D|} \mathbb{E}\left[\mathcal{L}(w_c; D_c)\right] \quad (1)$$

where, $\mathbb{E}\left[\mathcal{L}(w_c; D_c)\right]$ represents the empirical risk calculated using model w_c with data sampled from client c's training dataset D_c, $|D_c|$ is the sample number of client c's training dataset, and $|D|$ is the total size of training dataset.

Test-Time Mobile Data Shift and Model Generalization. Orthogonal to data heterogeneity *across* clients, the data distribution at the *same* mobile client may also vary between *train-* and *test-time* due to mobile users typically collecting environmental and user data in open and dynamic settings, a phenomenon known as *test-time data shift* [28], [59], [80]. Although such data shift is extensively studied in the generic machine learning literature, *e.g.*, model generalization [60], [61], it is largely overlooked in PFL, where the personalized model often "overfits" the local (training) data at each client. A few pioneer studies [28] exploit the global model during federated training to enhance the generalization of the personalized models to combat test-time data shifts. However, they may not function properly under extreme data heterogeneity (see Sec. 2.2).

2.2 Problem Definition

Due to the ubiquity of both data heterogeneity and test-time data shift in mobile applications, we explore *shift-robust PFL* with mobile devices as follows:

$$\min_{w_1, w_2, \dots, w_C} l(w_1, \dots, w_C) = \sum_{c=1}^{C} \frac{|D_c|}{|D|} \mathbb{E}\left[\mathcal{L}(w_c; \hat{D}_c)\right] \tag{2}$$

where $\{D_1, D_2, \dots, D_C\}$ and $\{w_1, w_2, \dots, w_C\}$ are training datasets and personalized models for clients $\{1, 2, \dots, C\}$, and client c holds D_c and deploys w_c. $\mathbb{E}\left[\mathcal{L}(w_c; D_c)\right]$ represents the empirical risk calculated using model w_c with data sampled from mobile client c's dataset which experiences data shift $\hat{D}_c$. $|D_c|$ is the sample number of client c's training dataset, and $|D|$ is the total size of training dataset.

We investigate the shift-robust PFL problem with the following scope.

- We care about *severe non-IID* (not independent, and identically distributed) label distributions across mobile clients. This is a challenging problem. For example, in sensor-based human activity recognition, a sports enthusiast is likely to gather human activity data about jogging and hiking, while a homebody may collect data about sitting and standing. Similarly, in Siemens factories that implement an FL system for learning defect detection models [64], the air conditioning production line might predominantly encounter cracks and scratches, while the razor line may primarily experience discoloration and deformation.
- We consider both the cases of *synchronous* and *asynchronous* clients. In both cases, we optimize both the training effectiveness (measured by accuracy on drifted data) and efficiency (measured by convergence time). The mobile clients are commodity mobile devices *e.g.*, Jetson Nano (commonly used in industrial control), which are capable of training small- to medium-sized models *e.g.*, MobileNet [63] and DistilBERT [72] via standard SGD, and the models can be reliably transferred between the mobile clients and the server.
- We focus on personalized federated training for typical mobile classification tasks, *e.g.*, image recognition [69], sensor-based human activity recognition [67], and text-based emotion detection [68]. Because the datasets in these mobile applications are highly *personalized* and exhibit *clusterability* due to user habits and lifestyles.
- We observe that the shifted labels of mobile client c are likely to have appeared in the training data of other clients [38]. Notable out-of-distribution (OOD) samples w.r.t. the entire federation is out of our scope. For instance, the homebody in the human activity recognition example may start jogging after the model is trained, while jogging is already seen from the training data of the sports enthusiast.

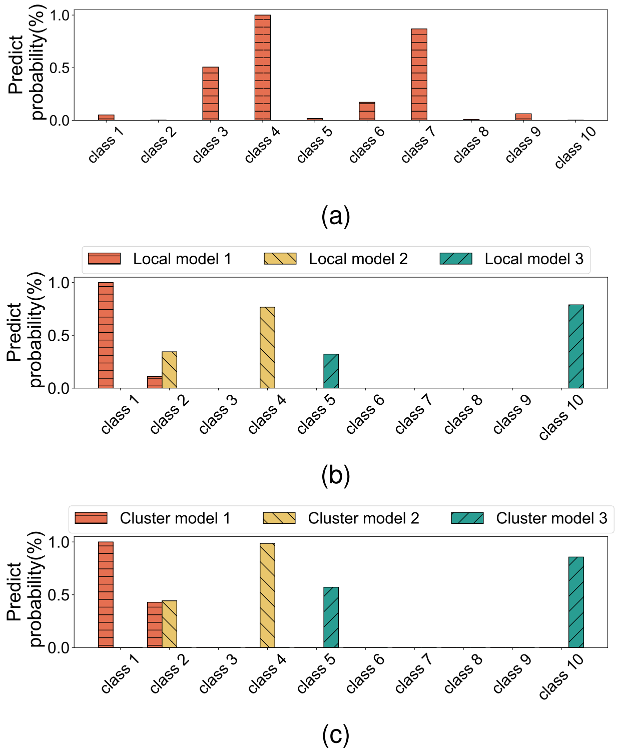


Fig. 2. Prediction probabilities of (a) the global model, (b) three local models and (c) three cluster models, for a sample with a true label of 4.

3 Mobile Shift-Robust PFL

This section explains the key idea (Sec. 3.1) to extract a generalized model in CFL even under extreme data heterogeneity, and presents an overview (Sec. 3.2) of ClassTer, our effective and efficient shift-robust PFL scheme.

3.1 Key Idea: Class-Wise Clustering

Our key novelty is the paradigm shift from the traditional *client-wise* clustering [16], [17], [53] to *class-wise* clustering, which allows *effective* aggregation of the cluster models into a generalized one via *knowledge distillation*. We utilize knowledge distillation rather than methods like parameter averaging [2] for its *efficiency*, *i.e.*, it only needs to run once at the server without excessive client-server communications.

Effective knowledge distillation always demands high-confidence soft labels [54]. However, the client-wise clustering scheme in conventional CFL often leads the cluster models to yield *wrong confidence* estimates, even though the

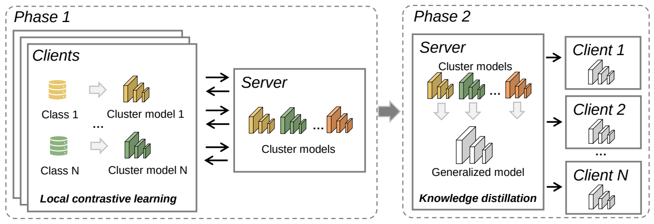


Fig. 3. Workflow of ClassTer.

output class is correct. To illustrate the reason, Fig. 2 plots the confidence (estimated by prediction probabilities) for a sample with the true label of 4, using three models: a **global model** trained on the full dataset, **local models** trained on local data, and **cluster models** obtained by conventional CFL. Compared to an ideal global model that delivers correct predictions, local models 1 and 3, along with cluster models 1 and 3, yield incorrect predictions. This is due to their training with a partial view (at each client or cluster), where the data are personalized and non-IID across clients or clusters, thus introducing large ***biases*** toward ***frequent*** classes in the training data rather than the correct ones.

To address this problem, our key idea is that the *class-wise clustering* of mobile clients in CFL can avoid the **class-specific biases** by *decoupling* the training of classes. Concretely, we learn each class independently, such that all per-class models are already trained with a holistic view on the entire (federated) dataset before aggregating into the generalized model. We also adopt contrastive learning [66] at mobile clients to avoid over-confidence in incorrect predictions in training the single-class models.

Additional Scalability Benefits. Despite the above-mentioned benefits of class-wise clustering in CFL, it raises concerns on whether it scales to practical classification tasks, which contain hundreds. We argue that class-wise clustering is plausible in mobile applications because mobile clients often only possess samples from a limited number of personalized classes. Additionally, a model trained to recognize a single class is more lightweight than one trained to recognize all classes (*e.g.*, hundreds), making it highly suitable for deployment and learning on mobile devices. We empirically validate the scalability and efficiency of our class-wise clustering mechanism in Sec. 6.3.5.

3.2 Solution Overview

Upon the above-mentioned principle of *class-wise clustering*, we develop ClassTer, an effective and efficient shift-robust PFL scheme for mobile devices. Algorithm 1 and Fig. 3 illustrate the workflow of ClassTer. First, the server initializes N cluster models, where N is the number of classification classes. In each iteration, the server assigns a cluster model with the slowest progress to the mobile clients to balance training latency across clusters. Each mobile client performs local contrastive learning on the downloaded cluster model, and uploads the local model to the server. The server aggregates the uploaded local models with the cluster model of the corresponding class. This iteration continues until all cluster models converge. To extract the generalized model, ClassTer distills knowledge from the cluster models. Finally, the generalized model is sent to all mobile clients, and each

Algorithm 1 ClassTer Training Procedure

```
1:  Initialize: Server predefines $N$ models $\{w_1, \ldots, w_N\}$ for $N$ task classes and generalized model $w_g$
2:  while not converged do
3:      for each client $c$ do
4:          Request single-class model $w_n$ for $n$-th class of local data.
5:          Local contrastive training $w'_n \leftarrow w_n + \eta \nabla f_n(w_n)$.
6:          Send updates $w'_n$ to the server.
7:      end for
8:      Aggregate updates for each model $n$.
9:  end while
10: Distill knowledge from $\{w_1, \ldots, w_N\}$ into generalized model $w_g$
11: Personalize the generalized model $w_g$ for each client denoted as $\{w_g^c\}$.
12: Output: Generalized and personalized models $w_g, \{w_g^c\}$
```

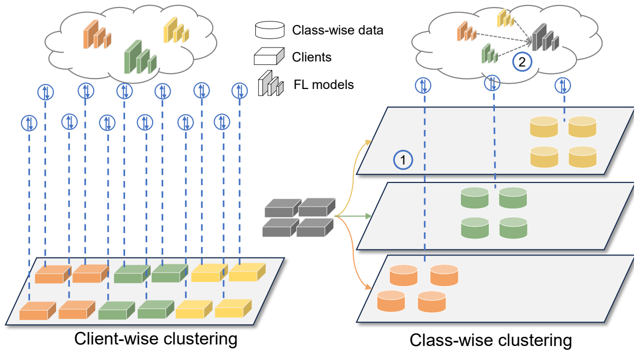


Fig. 4. Comparison between client-wise and class-wise clustering.

mobile client fine-tunes the output layer of the generalized model to obtain a personalized model. During inference, based on the occurrence of a mobile data shift measured by the entropy of output logits, each mobile client adaptively selects either the generalized or personalized predictions as the output. As next, we explain the detailed designs of ClassTer in both the synchronous and asynchronous settings.

4 ClassTer with Synchronous Clients

This section presents the key modules to enable shift-robust PFL with synchronous clients. As shown in Fig. 4, ClassTer first *fairly schedules* clients to *contrastively train* multiple single-class models. Then, the server extracts a generalized model from these single-class models via *knowledge distillation* to enable shift robustness.

4.1 Local Contrastive Learning

Why Contrastive Learning. As mentioned in Sec. 3.2, we leverage contrastive learning to at clients to improve the accuracy and avoid over-confidence of the single-class models. Contrastive learning is more suitable for extreme non-IID data distribution than supervised learning, as it allows a single-class model to learn only from the target class. For example, one-class classification (OCC) methods [74], [75] are initially contrastively trained with only one class of data which is one of the most extreme non-IID data distribution. During testing, OCC classifies samples by comparing their features to those of the training samples.

ClassTer follows the contrastive learning method in OCC, but at test time, the OCC classification method is *infeasible* because using training sample features would compromise privacy in FL. Therefore, ClassTer trains the single-class model to learn the features of the target Concretely, the objectives of local training include the following. *(i)* To learn accurate target class features, ClassTer *minimizes* the distance between augmented representations of the same sample and *maximizes* the distance between different samples. *(ii)* To classify target and non-target data, ClassTer *maximizes* confidence within the target class and *minimizes* confidence outside the non-target class.

Loss Design. According to the principles above, we devise a training loss that utilizes four data samples from the local dataset $\mathcal{D}m$ of client m: $\xi_{+,p}, \xi_{+,q}, \xi_{-}$, where ξ_{+} is the sample within the data distribution, and ξ_{-} is the sample outside the distribution. Specifically, the loss is defined as:

$$\mathcal{L} = \mathcal{L}_{con} + \mathcal{L}_{classify} \tag{3}$$

where $\mathcal{L}_{con}$ is the standard contrastive loss for learning features of target class data:

$$\mathcal{L}_{con} = -\log \frac{\exp(\text{sim}(w(\xi_{+}), w(\hat{\xi_{+}}))/\tau)}{\exp(\text{sim}((w(\xi_{+,p}), w(\xi_{+,q}))/\tau)} \tag{4}$$

$\{\xi_{+,p}, \xi_{+,q}\} \in \xi_{+}$, $\hat{\xi_{+}}$ is the augmentation (*e.g.*, crop) of ξ_{+}, and sim is the cosine similarity and τ is a temperature hyper-parameter.

For $\mathcal{L}_{classify}$, it naturally transfers the feature comparison method from OCC to FL. This involves a predefined feature center v_c, minimizing the distance between the target class features and v_c, and maximizing the distance between the non-target class features and v_c:

$$\log \frac{exp(sim(w(\xi_{+}), v_c)/\tau)}{exp(sim(w(\xi_{-}), v_c)/\tau)} \tag{5}$$

However, calculating the distance between the feature and v_c produces a much larger gradient than $\mathcal{L}_{con}$, resulting in a trivial solution where all features collapse to the predefined center v_c. Therefore, we shift to training a classifier to classify the learned features instead of directly comparing the features.

$$\mathcal{L}_{classify} = \sum_{\{\xi_{+}, \xi_{-}\} \in \xi} \text{CrossEntropy}(w_{classifer}(w(\xi))) \tag{6}$$

Local Training Workflow. In each iteration, the client will take a batch of data D_B. Based on the target data class, the client splits D_B into ξ_{+}, ξ_{-}. For ξ_{+}, the client generates $\hat{\xi_{+}}$ by applying random augmentations. For the most extreme non-IID data distributions, where each client only holds one class of data, we can generate Gaussian noise as ξ_{-}.

4.2 Inter-Cluster Scheduling

Imbalanced Convergence of Cluster Models. Although local contrastive learning at the mobile client boosts the accuracy of the cluster models *i.e.*, the single-class models,

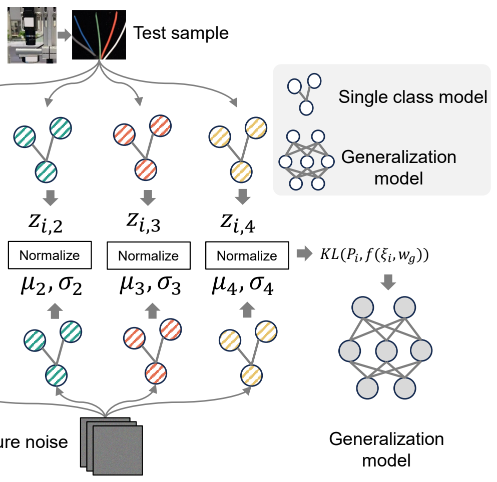


Fig. 5. Illustration of knowledge distillation from cluster models into a generalized model.

these models may not converge simultaneously even with synchronous clients. This is because each mobile client is involved in local training *multiple* cluster models, and the local training of these models will *compete for the limited resources* at the client. A cluster model trained with few clients per iteration suffers from a long convergence time. And the imbalanced convergence time causes fast cluster models to wait for the slow ones for knowledge distillation, leading to prolonged overall training latency.

Cluster Model Scheduling. ClassTer strategically selects cluster models per FL iteration for local training to roughly synchronize the convergence of all cluster models. Formally, let E_{max} be the number of local iterations still required by the cluster model with the longest convergence time, and $\hat{E}$ be the average number of local iterations across all cluster models. We aim to minimize v, the convergence time imbalance index across clusters defined as $v = (\frac{E_{max}}{\overline{E}} - 1)$. Although E_{max} and $\overline{E}$ reflect the number of local iterations still needed, they are difficult to measure. Therefore, we convert the number of remaining local iterations into the number of completed local iterations, which is easier to assess. Specifically, assume all cluster models need the same number of local iterations, denoted as $\hat{E}$, and let the number of local finished iterations for the $n-th$ cluster model be e_n. Then E_{max} can be calculated as $\hat{E} - \text{Min}(e_n)$, where $\overline{E} = \frac{1}{N}\sum(\hat{E} - e_n)$. Consequently, v is expressed as:

$$v = 1 - \frac{\text{Min}(e_n)}{\overline{e_n}} \tag{7}$$

Then our goal becomes maximizing the minimum proportion of effective training rounds, $\text{Min}(e_n)$. Thus, for each round, ClassTer always assigns clients to the cluster with the lower e_n.

4.3 Generalized Model Extraction

As mentioned in Sec. 3.2, we extract the generalized model from the single-class models via knowledge distillation (KD) at the server. However, naive KD may fail due to the *misaligned objectives* between the generalized model and the single-class models. Note that KD implicitly assumes that both the teacher and the student are intended for the same classification task [54]. Yet in our case, the generalized model aims to recognize multiple classes whereas each single-class model targets at only one class. The different objectives makes the soft labels between the generalized and the single-class models incomparable, which may lead to sub-optimal or even wrong knowledge transfer.

To align the objectives between the generalized model from single-class models, we use the output probabilities from each single-class model as the predicted probabilities for one class in the generalized model and create soft labels for training the generalized model. However, differences in prediction scales between single-class models still cause problems. For example, if one model's probabilities are between 1 and 10 and those of another are between 0 and 1, the soft labels will always favor the former's probabilities over the correct ones. Therefore, normalizing the probabilities of single-class models is necessary. We observe that the dimensional differences in single-class models are data-independent, meaning that regardless of the data provided, the model's output remains at the same scale. Therefore, we can obtain the necessary statistics (*e.g.*, mean and standard deviation) for normalizing the model's probabilities by using some pure noise data.

Fig. 5 illustrates the process to compute the distilling probabilities from the single-class model output for a sample ξ_i. Firstly, each single class model(w_n) calculates the mean (μ_n) and variance (σ_n) of its normalized output from a set of pure noise data. For a given sample ξ_i in the dataset $\mathcal{S}$, we generate the logits output $z_{i,n} = f(\xi_i, w_n)$ using single-class model $w_n \in w_{1\ldots N}$. Then, we use the mean and variance to normalize the model logits output and combine the normalized result as:

$$P_{i\|i=n} = \frac{z_{i,n} - \mu_n}{\sigma_n} \tag{8}$$

In this context, P_i reflects the accurate inference probability distribution for the sample ξ_i. Simultaneously, P_i shares the same vector space as the prediction probability distribution ($f(\xi_i, W_g)$) of the generalization model W_g. Therefore, the final distillation objective is formalized as follows:

$$\mathcal{L}(W_g, W_{1\ldots N}) = \mathcal{KL}(P_i, f(\xi_i, W_g)) \tag{9}$$

Since all knowledge distillation occurs on the server, the mobile devices have no additional overhead. Additionally, the extra overhead can be disregarded due to the server's powerful computational capacity.

5 Supporting Asynchronous Clients

This section extends ClassTer to asynchronous clients for faster convergence (measured by wall clock time), where clusters aggregate updates as soon as they are available from each client. Supporting asynchronous clients makes ClassTer more practical because clients in real-world FL systems are likely to experience heterogeneous resource availability and transmission bandwidth, causing them to upload local model updates asynchronously to the server. In this context, using synchronous aggregation for asynchronous updates results in long waiting times and inefficiency. However, asynchronous model aggregation demands dedicated *staleness control* to retain training convergence and model accuracy [40]. ClassTer harnesses the notion of critical learning

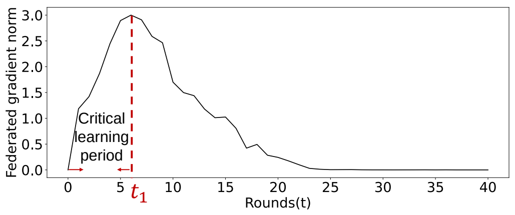


Fig. 6. Critical learning period and training rounds.

periods (Sec. 5.1) and performs both intra-cluster (Sec. 5.2) and inter-cluster (Sec. 5.3) scheduling to support effective and efficient asynchronous CFL, as explained below.

5.1 Critical Learning Period in FL

The effectiveness of asynchronous model aggregation is often hampered by the stale updates of local models [49], [50]. To mitigate the impact of staleness, updates from lagging clients can be either discarded [50] or decayed [40]. We adopt the latter since slow clients may contain important personalized data, and thus excluding them from training may drastically deteriorate the model accuracy [16].

In addition to *which* mobile clients to include, an equally important yet often overlooked issue is *when* to include these clients for training. Pioneer studies [34], [35] show that excluding important mobile clients from the *critical learning periods* (often the early stage) of FL can lead to irreversible accuracy degradation. Thus, ensuring access to comprehensive data during the critical learning period is essential to achieve high model accuracy in federated learning.

Probing Critical Learning Period. Existing methods identify the critical learning periods in the *synchronous* settings based on the Federated Gradient Norm (FGN) [35]. $FGN(t) = \sum_{k\in M^{(t)}} \frac{N_k}{\sum_{k\in M^{(t)}} N_k} \Delta L_k^t$, where t denotes the current round of FL training, and $M^{(t)}$ represents the set of all devices participating in that round t. FGN calculates the average training loss across these clients for each round. Specifically, for a probabilistic classification model $p_w(y \mid x)$, where w is the model parameter, and $L(x, y; w)$ denotes the loss function computed for input x and label y, the gradient of the loss for the example (x, y) is $g(x, y; w) = \frac{\partial}{\partial w} L(x, y; w)$. After gradient descent for the (x,y) sample, the training loss is computed as $\Delta L = L(x, y; w - \eta g(x, y; w)) - g(x, y; w)$, where η is the learning rate. The FGN quantifies the correlation between the critical learning period and training rounds, as illustrated in Fig. 6. The rounds $(0, t_1]$ are identified as the critical learning period, determined by the threshold-based criterion: $\frac{FGN(t)-FGN(t-1)}{FGN(t-1)} \geq \delta$. If this criterion is met, round t is a part of the critical learning period.

We extend the notion of critical learning periods to the *asynchronous* setting and propose an *asynchronous FGN (AFGN) metric* to account for client model training epoch p at time t. Specifically, we follow the idea of FGN to identify the mutation point of the gradient norm and further refine the measurement to include time-variant factors. It quantifies the difference between the gradient norm of the $n-th$ single-class model at the time t.

$$AFGN(p(t))_i = AFGN(p(t-1))_i + \frac{N_k}{N} \times \Delta L_k^T \tag{10}$$

where N is the total sum of training data across all clients in iteration t, dynamically changing due to system heterogeneity. N_k indicates the current data volume for k, and ΔL_k^T is the training loss uploaded by client k in round T.

As AFGN is an iterative function, we use the second derivative of AFGN $\frac{d^2}{dt^2}\text{AFGN}(t)$ to judge the critical learning period of the n-th cluster as follows:

$$\begin{aligned} Trps(Clusters^{(n)}) &= \frac{d^2}{dt^2}\text{AFGN}(t) \\ &\approx \text{AFGN}(t) + \text{AFGN}(t-2) - 2\text{AFGN}(t-1) \end{aligned} \tag{11}$$

where $Trps$ denotes the learning period in the n-th cluster. We monitor $Trps$ at time t for each cluster. When it surpasses a specified threshold, the cluster is considered to be in a critical learning period. The higher the $Trps$ value, the greater the criticality of the learning period.

Quantifying the critical learning period allows for more balanced intra-cluster aggregation in its early stages. In each KL training round, more clients can be aggregated into a few single-class models at the inter-cluster level.

5.2 Asynchronous Intra-Cluster Scheduling

Unlike synchronous intra-cluster clients with uniform update frequencies, asynchronous intra-cluster clients exhibit varying update frequencies, causing staleness. The staleness significantly reduces learning accuracy during the critical learning period. Therefore, to mitigate the staleness issue, we schedule the weight aggregation ratio according to the critical learning period.

Aggregating asynchronous updates with staleness into single-class models equally can degrade accuracy and extend convergence times. Specifically, aggregating stale updates during critical learning periods may cause irreversible accuracy loss. To optimize the trade-off between accuracy and latency, ClassTer strategically schedules aggregation weights among client updates to maximize updates during critical learning periods for improved early-stage results, while limiting updates towards the end to enhance federated learning efficiency.

Initially, we predict the average training time per round (t_{avg}) and its variance (σ) through local testing with multiple clients in the first round, and these metrics are updated iteratively during the FL process. When a client i submits an update, it uploads the local model w_{local}^i and its training time t. Based on the asynchronous critical learning period prob, we modify the Hinge aggregation formula in [40] as:

$$\beta = \begin{cases} \alpha & \text{if } t < T_{avg} + 3 * \frac{AFGN(i)}{AFGN_{max}}\sigma \\ \frac{\alpha}{a(t-\tau-b)+1} & \text{otherwise} \end{cases} \tag{12}$$

where α is set to 0.5, a to 10, and b to 4, as used by [40]. The uploaded model w_{local}^i is then aggregated into the single-class model w_{server}^i by $w_{server}^{i+1} = \beta w_{local}^i + (1-\beta) w_{server}^i$.

5.3 Asynchronous Inter-Cluster Scheduling

In the synchronous setting, ClassTer simultaneously assigns multiple mobile clients to a cluster that has been trained with a few mobile clients. However, in the asynchronous setting, clients are assigned to clusters on arrival. When clusters are trained with similar numbers of clients, each cluster might receive balance but few clients at the same time for training, impacting the training accuracy of the single class model during critical learning periods.

ClassTer further introduces an asynchronous scheduler that dynamically adjusts client assignments to align training times across various clusters, complementing the synchronous strategy described in Sec. 4.2. It tracks the count of updates for each single-class model with $\{e_1, e_2, \ldots, e_N\}$, where e_n indicates the number of updates for the n-th single-class model. Under conditions where clients can train multiple clusters, we prioritize assigning clients to contribute to single-class models with fewer updates, adhering to the principle of minimizing bottlenecks [76]. The reason is that single-class models with fewer updates can prioritize and capture more clients for training, thereby speeding up the training of slower single-class models. However, the immediate updates to $\{e_1, \ldots, e_N\}$ result in each single-class model being trained by only a few clients simultaneously, leading to performance decline during critical learning periods. This occurs because these updates frequently change the priority of the single-class models, causing clients to be dispersedly allocated rather than assigned in batches. Thus, updates are not immediately reflected in $\{e_1, \ldots, e_N\}$; instead, counts are cached for K updates before a client assignment adjustment. This strategy of delayed updates allows clients to concentrate on training specific single-class models within a brief period, minimizing gradient errors potentially caused by limited client participation during critical learning periods.

6 EXPERIMENT

6.1 Experiment Setup

Implementation. We implement ClassTer using Python 3.9 and PyTorch 1.10 for the server and mobile clients, respectively. The server is equipped with two RTX 3090 GPUs and 128GB RAM. We use 20 mobile and embedded devices of five types: Jetson Nano C_1, Jetson NX Xavier C_2, Jetson Nano Orin C_3, Jetson AGX Xavier C_4, and Raspberry Pi 4 C_5. They represent diverse computing capabilities and form a distributed FL system.

Tasks, Datasets, and Models. We experiment with four real-world mobile applications. And the data assigned to each client is Non-IID and unbalanced. We simulate the degree of Non-IID using the Dirichlet distribution [82], with the hyperparameter set between 0.1 and 0.5. A value of 0.1 represents the most extreme Non-IID data distribution, while 0.5 represents nearly IID data distribution.

- **Image Recognition** (T_1) is ubiquitous in smart cameras/robots. We employed the CIFAR-10 dataset [82]. To demonstrate scalability, we also utilized the CIFAR-100. We utilize Resnet-8 for this task.
- **Human Activity Recognition**, HAR (T_2) on mobiles has gained significant attention [17]. We adopt the HARBox dataset [17], which comprises sensor data from 121 users. The model utilizes the LSTM architecture.
- **Text-based Emotion Detection** (T_3) is deployed on mobile devices to detect user emotion. We adopt DailyDialog [71] dataset, which has 13,000 dialogue samples encompassing seven different emotions. We utilize the transformer-based model DistillBert [72] for this task.

- **Image aesthetics evaluation** (T_4) evaluate the aesthetics of user-taken photos. We gathered 1,000 images from 20 users, classifying them into five aesthetic value categories. We use ResNet-8 for this task.

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 the software. In detail, we simulate 50% fast devices and 50% slow devices. For real-world experiments T_4, we conducted them with $3C_1$, $5C_2$, $4C_3$,$2C_4$ and $6C_5$.

Baselines. We adopt five mainstream FL algorithms with mobile devices as performance comparison baselines. These baselines offer SOTA methods for evaluating system efficiency and model accuracy in extreme Non-IID data distributions, both before and after data shift They are configured as follows:

- **Standard FL methods:** These methods provide accuracy and training time baseline since they are not optimized for any particular mobile scenario.
  - FedAvg [2]: the server calculates the average of all mobile clients' weights as a global model and then deploys the same global model to all clients.
- **Personalized FL (PFL) methods** provide high accuracy results without data shift as the model is personalized for each client.
  - Cluster-based PFL: IFCA [53]: is a learning personalized models method, which learns different models for each cluster to realize personalization.
  - Fine-tuning based PFL: FedAvg+Fine-tuning [73] fine-tunes the global model at the client to realize personalization.
- **Shift-robust PFL** mitigate the accuracy decline brought by test-time mobile data shift.
  - DM-PFL [28] trains generalized and personalized models simultaneously and adaptively select output to address data shift.
- **Asynchronous FL methods:**
  - FedAsyn [40] promptly aggregates the model and distributes updates to clients in an asynchronous manner.

6.2 Performance Comparison

We test ClassTer and four baseline methods (*i.e.*, FedAvg, IFCA, FedAvg+FT, DM-PFL, and FedAsyn) across Image recognition(T_1), Human activity recognition (T_3), and Text-based emotion detection (T_3) tasks under extreme device heterogeneity (*i.e.*, Slow devices have 20 times the local training time of fast devices.) and non-IID data distribution (*i.e.*, Dirichlet distribution with a hyperparameter of 0.1 [82]).

Fig. 7 presents the results. First, ClassTer demonstrates the highest accuracy across all three tasks. For instance, as shown in Figure Fig. 7b, our method outperforms the five baselines by 10%-33% in accuracy. This is because ClassTer effectively extracts a global model from single-class models trained through class-wise clustering, reducing the accuracy decline caused by data shift. Second, ClassTer also exhibits the fastest convergence speed across the three tasks. As depicted in Figure 4b, ClassTer achieves the highest accuracy with the fewest number of training rounds compared to the four baselines. This is because ClassTer balances the training progress of single-class models through intra-cluster scheduling and alleviates the staleness issue from slow devices with critical learning-based inter-cluster scheduling.

Summary: ClassTer achieves an optimal trade-off between training time and accuracy benefiting from its critical learning period-based intra- and inter-cluster scheduler. In extreme shift scenarios, ClassTer outperforms the a-/synchronous PFL baselines by 1%-34% in accuracy and reduces the training time by 21%-91%. This makes ClassTer a promising solution for federated learning in mobile applications with prevalent extreme data and system heterogeneity.

6.3 Performance in Various Scenarios

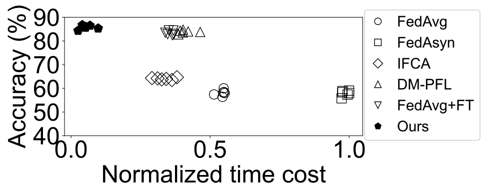
6.3.1 Performance under Different Data Shifts

We test ClassTer and four baseline methods across six shift scenarios with the T_1 task, utilizing three different numbers of FL rounds. In mobile scenarios, the degree of data shift can vary due to the changing of deployment environments (*e.g.*, stable environment results in 0% data shift, and an extremely dynamic environment results in 100% data class shift). Fig. 8 shows the results. First, ClassTer exhibits the best accuracy result under different degrees of data shifts. For example, as shown in Fig. 8b, with 400 rounds FL training, demonstrates accuracy improvements of 4.04% $\sim$ 50.45% compared to non-shift-robust methods FedAvg, FedAvg+FT, ICFA and 5.16% $\sim$ 12.28% compared to shift-robust method DM-PFL. Second, ClassTer achieves shift-robustness most rapidly. For instance, as shown in Fig. 8a, with only 200 rounds of FL training, ClassTer achieves 78.74% and 36.5% accuracy with 0% and 100% data shift, which are 5.45% and 14.61% higher than shift-robust method DM-PFL. As shown in Fig. 8c, after 800 rounds of FL training, ClassTer still achieves an accuracy 1.66%-49.73% higher than the baselines under various data shift conditions.

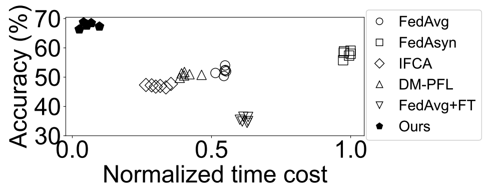
Summary: ClassTer establishes an effect shift-robust solution for extreme non-iid data distribution with several advantages: *i)* ClassTer achieves the highest test accuracy under various data shift conditions by aggregating single-class models into a generalized model. *ii)* ClassTer adapts to extreme non-iid data distribution through *class-wise clustering*, achieving the highest accuracy with the fewest rounds among FedAvg, FedAvg+FT, ICFA, and DM-PFL. This enables ClassTer to provide shift-robust deep learning models for mobile applications in the shortest time.

6.3.2 Performance over Different Non-IID Settings

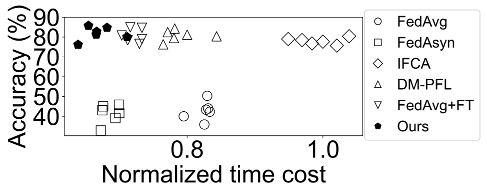
We test ClassTer and four baseline methods across five non-iid degree scenarios with the T_1 task. First, ClassTer exhibits the best accuracy result with extreme Non-IID data distribution. For example, as shown in Fig. 9e and Fig. 9f, ClassTer accuracy improvements of 14.34%-23.5% with shift and 1.66%-49.73% without shift. Second, ClassTer remains effective under general non-iid conditions. As shown in Fig. 9e and Fig. 9f, with a 0.5 Non-IID degree, which represents a weak Non-IID data distribution, ClassTer still



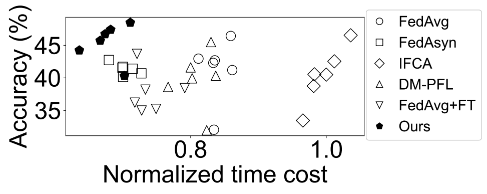
(a) Performance on T1 without data shift



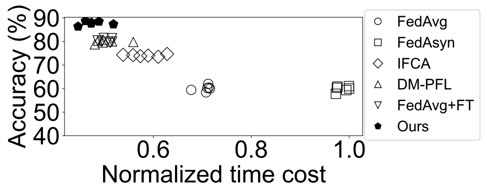
(b) Performance on T1 with data shift



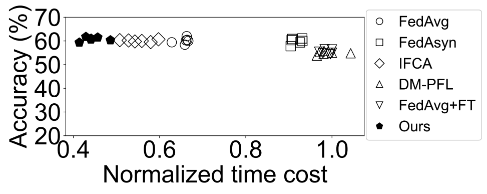
(c) Performance on T2 without data shift



(d) Performance on T2 with data shift

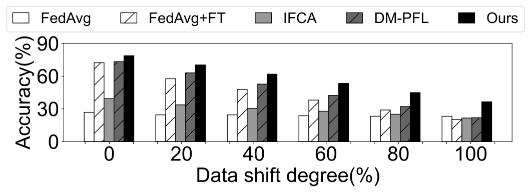


(e) Performance on T3 without data shift

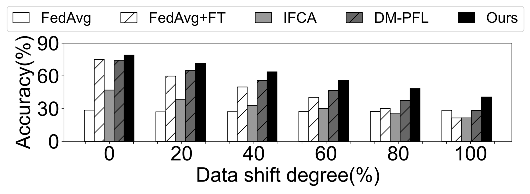


(f) Performance on T3 with data shift

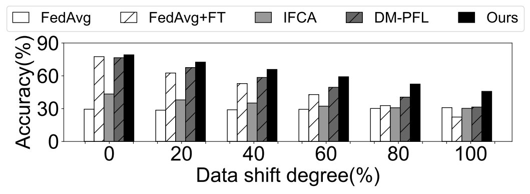
Fig. 7. Comparison of accuracy vs. training time between ClassTer and other baselines on diverse tasks.



(a) Accuracy with various shifts using 200 FL rounds



(b) Accuracy with various shifts using 400 FL rounds



(c) Accuracy with various shifts using 800 FL rounds

Fig. 8. Accuracy with different degrees of shift on CIFAR10 in Dir(0.1) non-IID setting.

achieves comparable accuracy (improving 0.12% without data shift and 2.45% with data shift compare with DM-PFL). Third, ClassTer provides higher accuracy more quickly across varying degrees of Non-IID data distribution. For instance, ClassTer achieves 1.55%-49.62% higher accuracy with 400 FL rounds(Fig. 9c and Fig. 9d) compared to baseline methods with 800 FL rounds(Fig. 9e and Fig. 9f) under 0.1 Non-IID data distributions.

TABLE 1
Quantity and performance ratios of fast and slow devices

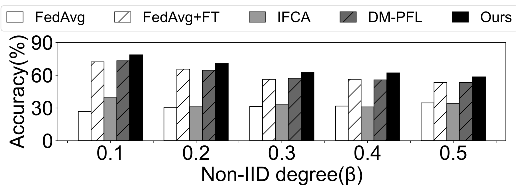
Index	Slow/Fast device quantity proportion	Slow/Fast device performance ratio
D1	1/1	1/1
D2	1/1	1/5
D3	1/1	1/10
D4	1/1	1/20

Summary: ClassTer is efficient in extreme Non-IID scenarios while remaining effective in general Non-IID conditions. The reason is that the *class-wise clustering* method enables effective FL training of single-class models with near-IID conditions under any data distribution.

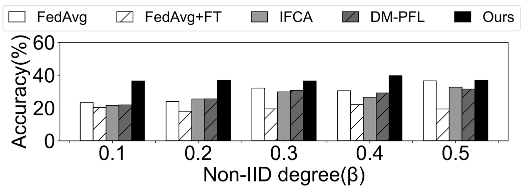
6.3.3 Performance on Different Mobile Devices

We test ClassTer and two baseline methods (*i.e.*, FedAvg and FedAsyn) under various heterogeneous device conditions in T_1 task to determine the highest accuracy achievable within the specified time frame. This reflects the superior capability of ClassTer in rapidly training models. In this experiment, clients are divided into fast and slow devices based on the time taken to complete each round of FL. The device ratio of fast and slow devices is 1:1, and the training time ratios for fast and slow devices to complete each round of FL are illustrated in the Tab. 1 as 1:5, 1:10, 1:15, and 1:20. Such device heterogeneity is common in mobile scenarios, for example, devices with GPUs can perform several times faster than those without GPUs. We record the training time taken by ClassTer and the baselines to achieve various accuracy levels within four hours. In some settings (*e.g.*, D4 and 30% accuracy), the baselines failed to achieve the target accuracy and were recorded as 0 (*e.g.*, FedAsyn in Fig. 10a). We evaluate accuracy on CIFAR-10 with 100% data shift.

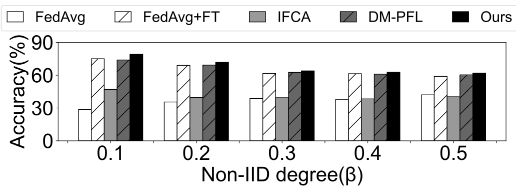
First, under varying heterogeneous device conditions, ClassTer consistently achieves the target accuracy in the shortest time. As shown in Fig. 10a, to achieve 30% accuracy,



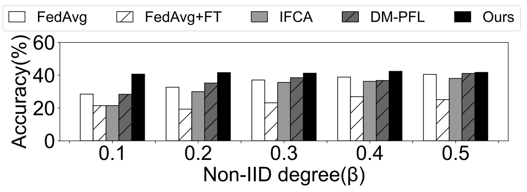
(a) 200 FL rounds and without data shift



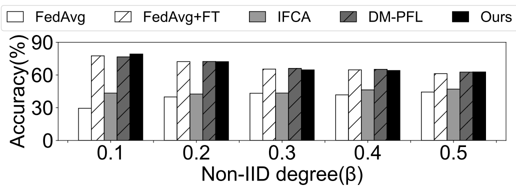
(b) 200 FL rounds and with 100% data shift



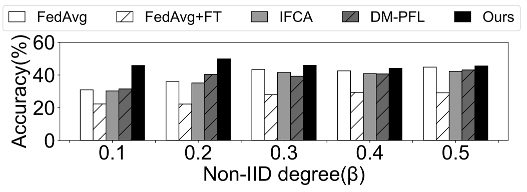
(c) 400 FL rounds and without data shift



(d) 400 FL rounds and with 100% data shift

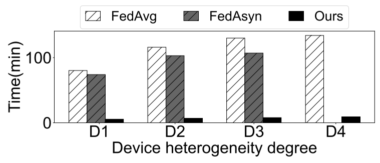


(e) 800 FL rounds and without data shift

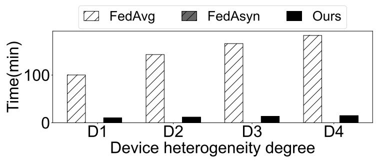


(f) 800 FL rounds and with 100% data shift

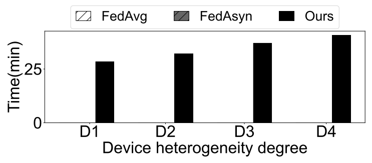
Fig. 9. Accuracy with different degrees of Non-IID data distribution on CIFAR10.



(a) Time for 30% accuracy



(b) Time for 40% accuracy



(c) Time for 50% accuracy

Fig. 10. Training time to achieve different accuracy.

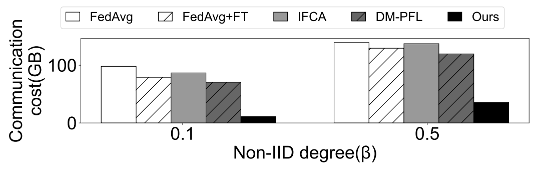
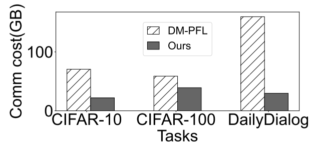
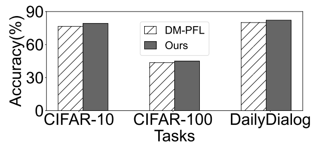


Fig. 11. ClassTer vs. other baselines in terms of communication cost.



(a) Communication cost



(b) Accuracy

Fig. 12. Comparison of communication costs and accuracy for various tasks with diverse scales.

ClassTer save up to 93.14% time compared to FedAvg and FedAsyn under 1:5, 1:10, and 1:15 device heterogeneity degree. Under 1:20 device heterogeneity degree, ClassTer saves 93.21% time compared to FedAvg while FedAsyn fails to achieve 30% accuracy due to the severe staleness issue. Second, ClassTer can achieve the highest accuracy within a limited time under various heterogeneous device conditions. As shown in Fig. 10c, within four hours, ClassTer reached 50% accuracy in as little as 28 minutes under multiple heterogeneous device conditions, whereas both FedAvg and FedAsyn failed to achieve 50% accuracy within the same four-hour period.

6.3.4 Communication Overhead

We compare ClassTer's communication cost with FedAvg, FedAvg+FT, ICFA, and DM-PFL in the image recognition (T_1) task. As shown in Fig. 11, ClassTer achieves 68.8%-77.6% communication cost reduction with the most extreme Non-IID scenario and 70.5%-74.6% communication cost reduction with the generic Non-IID scenario(Non-IID degree is 0.5). *The reason is class-wise clustering* reduces training complexity, thus requiring fewer communication rounds to reach the target accuracy.

6.3.5 Scalability to Transformer Models

We test ClassTer and DM-PFL across three different scale tasks (CIFAR-10, CIFAR-100, and DailyDialog) to validate ClassTer's scalability. As shown in Fig. 12, ClassTer achieved a reduction in communication costs by 33.3%-81.4% and an accuracy gain of 1.3%-2.66% without data shift across different tasks, validating its scalability.

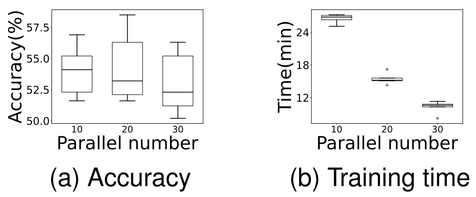


Fig. 13. Performance of ClassTer with different parallel device numbers.

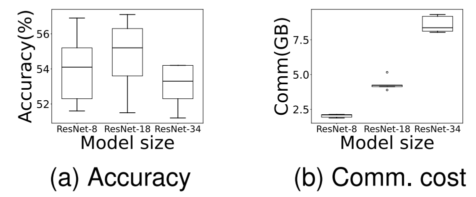


Fig. 14. Perfomance of ClassTer with different model size.

6.4 Micro-benchmark

6.4.1 Impact of Parallel Number

We evaluate the training time and accuracy with 100% data shift using different parallel numbers in ClassTer (*i.e.*, 10, 20, and 30), which represent the scalability of ClassTer to a large FL system. As shown in Fig. 13, with the increase of parallel number, ClassTer can reduce up to 61.5% training time to achieve similar FL accuracy. This is because updates between different single class models do not interfere, enabling more clients to train simultaneously.

6.4.2 Impact of Single-class Model Size

We assess the communication cost and accuracy with 100% data shift of ClassTer using various sizes of single-class models. This shows that ClassTer can lower the overhead in the federated phase by reducing the size of the single-class model. As shown in Fig. 14, using the smaller model, ClassTer achieves 73.3% communication cost reduction and similar accuracy.

6.4.3 Impact of Local Training Round

We test the communication cost in PFL for achieving 50% accuracy under a 100% data shift (where the test data is completely different from the training data) using different local training rounds in ClassTer. This indicates that ClassTer can reduce communication cost by extending local training rounds. As shown in Fig. 15, with the increase of local training rounds, ClassTer can reduce up to 41.64% communication cost. This is because, after class-wise clustering, clients training the same single-class model share similar optimization objectives, so more local training rounds do not lead to significant optimization conflicts.

6.4.4 Impact of knowledge distillation phase

We tested the percentage of time taken by the knowledge distillation(KD) phase and the federated learning (FL) phase during the convergence of ClassTer under four heterogeneous device conditions. The experiment showed that the additional time overhead introduced by knowledge distillation is acceptable. As shown in Tab. 2, the time spent in the KD phase under these four conditions ranged from only 5.17% to 7.39%, accounting for only a small fraction of the total time overhead. Since the KD-phase occurs entirely on the server, the time overhead is stable and manageable, making it acceptable for mobile systems.

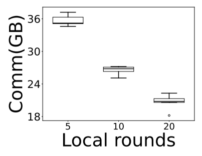


Fig. 15. Communication cost of ClassTer with different local training rounds.

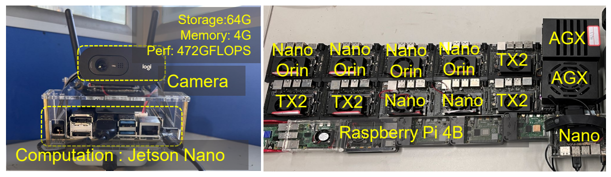


Fig. 16. Illustration of the case study with 20 mobile devices.

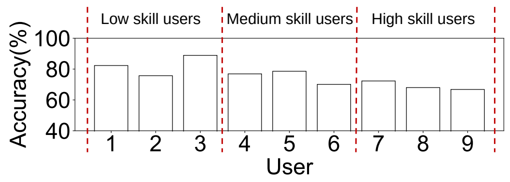


Fig. 17. Illustration of personalized mobile data caused by diverse user photography habits and skills.

6.5 Case Study

We use twenty mobile devices for a two-day study with the aesthetics evaluation app, which helps users automatically evaluate the aesthetics of their photos. Twenty participants labeled each image sample collected locally from their mobile phones from "1-5" based aesthetics, resulting in imbalanced data across devices due to varying photography skills. The system is deployed on 20 devices(as shown in Fig. 16) and each device holds one user's data.

To test ClassTer's performance under different shifts, we have three group users with varying photography skills (low, medium, and high) using the model trained by a user with low photography skills to conduct photo tests and assess accuracy. As shown in Fig. 17, the model achieves the highest accuracy when evaluating photos taken by low-skill users. For photos taken by medium-skill and high-skill users, accuracy decreased due to the data shift from improved shooting skills. However, ClassTer's class-wise clustering maintained shift robustness, resulting in a limited accuracy drop.

7 Related Work

Generic and Personalized FL for Handling Non-IID Data. Generic FL aims to train a *single* global model to serve all clients, with FedAvg [2] as the standard method. However, *data heterogeneity* presents challenges. To address this, prior efforts enhance FedAvg on the client side, *e.g.*, by regularization [7], [8], [36] and hyperparameter control [10], [47] to mitigate gradient divergence from non-IID data, or on the server side, *e.g.*, by weight matching [45] and knowledge distillation [43], [44]. Nonetheless, the diverse data distributions among clients make it challenging to fit a single global model to each client's specific data [12], [55].

TABLE 2
Quantity and performance ratios of fast and slow devices

	KD phase	FL phase	Overall
D1	7.39%	92.61%	100%
D2	6.56%	93.44%	100%
D3	5.68%	94.32%	100%
D4	5.17%	94.83%	100%

Personalized FL (PFL) produces *multiple personalized models* to fit local data distributions and address non-IID mobile data issues. PFL can be divided into two types based on the presence of a global model. *First*, without a global model, PFL methods directly train *multiple* personalized models, each tailored to a specific data distribution. *Second*, with *a global model*, the PFL architecture is "FL training + personalizing the global model." Methods for personalizing the global model include local fine-tuning [43], [73], meta-learning [14], [15], [18], and model interpolation [36], [37]. For example, Fallah *et al.* propose a meta-learning method [15] where a global meta-model is learned during FL training and each client fine-tunes it locally. However, relying on a single global model can degrade personalized model performance due to difficulties in converging across diverse local data distributions [12].

Therefore, maintaining *multiple global models* in PFL for better personalization emerges, employing techniques like clustering [12], [16], [17], [83], [84], [85] and multi-task learning [21], [22]. For example, ClusterFL *et al.* [17] proposes a cluster-based PFL method where the server identifies the cluster structure using KL-divergence between the uploaded models. Clients with similar data distributions are grouped into clusters, each maintaining its own global model. Thus, ClassTer follows the cluster-based PFL scheme, leveraging the high clusterability in mobile application data distributions despite their extreme non-IID nature.

FL with Data Shift. Data shift is a common issue, which occurs when test data distribution diverges from training data, risking accuracy [28], [38], [87]. To address this, generic FL enhances shift robustness typically in two ways, *i.e.*, on-device adaptation [46] and improving generalization during the training of global model [28], [39]. While cluster-based PFL effectively manages *extreme* non-IID data, it is less resilient to data shifts in mobile deployments. Integrating the above shift-robust methods with cluster-based PFL in mobile settings presents challenges. In particular, when integrating cluster-based FL with on-device adaptation methods, cluster center models fail to adapt to data distributions that experience shifts, as these models overfit the original data distribution. Another approach to achieving data shift robustness in cluster-based FL is to use the output of a generalized model to correct the output of the cluster center model. However, due to extremely non-IID data distributions, existing FL methods fail to train a generalized model effectively. To achieve shift robustness in cluster-based PFL, ClassTer focuses on enhancing the global model's generalization, reducing additional client training resource costs.

Synchronous and Asynchronous and FL. Device heterogeneity refers to the diversity in computational resources [93]. It can be addressed by modifying the model architecture and system-level adaptation. The former assigns lightweight models to devices with lower resources [51], [88]. For example, Li *et al.* [52] utilize knowledge distillation to aggregate models. However, for clients with lower resources but abundant data, assigning lightweight models may impair the extraction of that portion of knowledge. The latter mainly fall into client selection-based methods [10], [47], [48], asynchronous FL (AFL) [16], [40], and semi-asynchronous FL (SAFL) methods [49], [50]. *First*, client selection-based methods [47], [48] select devices with similar performance during each round of FL to avoid waiting issues. For example, FedBalancer [10] optimizes the selection of clients based on both device capacity and data distribution, thereby enhancing convergence speed. *Second*, adaptive model aggregation or asynchronous FL [29], [30], [31] controls staleness by scheduling model updates. Specifically, ArtFL achieves similar local training times across different devices by adjusting the amount of training data. *Third, adaptive weight dropping*, or semi-asynchronous FL [33], [49], [50], discards extremely stale models. However, these methods overlook critical learning periods, leading to significant degradation. In mobile scenarios, selecting clients with similar performance during each round in FL is infeasible. Therefore, ClassTer adopts asynchronous methods to optimize efficiency.

Existing methods [34], [35] demonstrate that during **critical learning periods** in FL, gradient errors introduced by staleness issue in semi-/asynchronous FL can impair the model's learning ability irreversibly. ClassTer embraces the asynchronous paradigm for its latency benefits and addresses aggregation challenges from stragglers and irreversible damage during critical learning periods by employing critical learning period-aware adaptive synchronization frequency schedule methods.

8 CONCLUSION

Due to the ubiquity of data heterogeneity and test-time data shift in mobile applications, we present ClassTer, a *shift-robust personalized FL* using class-wise clustering on mobile devices. First, we present a paradigm shift from traditional *client-wise* clustering to *class-wise* clustering for combating test-time mobile data shifts, which allows the effective aggregation of cluster models into a generalized one via knowledge distillation. Second, we extend ClassTer to support *asynchronous* mobile clients for faster convergence, measured by wall clock time, where clusters aggregate updates as soon as they are available from each mobile client. Supporting asynchronous mobile clients enhances ClassTer's practicality, addressing the heterogeneous resource availability and dynamic network bandwidth in real-world FL systems. By harnessing *critical learning periods* and performing both intra- and inter-cluster scheduling, we control staleness issues and optimize convergence in asynchronous shift-robust PFL. Evaluations on diverse popular mobile tasks and real-world scenarios over twenty mobile devices show that ClassTer achieves training time reductions of up to 91%, accuracy improvements of up to 50.45%, and communication cost reduction of up to 74.6%. In the future, we plan to expand ClassTer to handle more complex tasks by using hierarchical federated learning to improve class-wise clustering performance and distribute network traffic, making it adaptable to various network environments.

ACKNOWLEDGMENT

This work was partially supported by the National Science Fund for Distinguished Young Scholars (62025205) the National Natural Science Foundation of China (No. 62032020, 62102317,62472354), and CityU APRC grant No. 9610633.

REFERENCES

[1] Yang, L., Huang, J., Lin, W., & Cao, J. (2023). Personalized federated learning on non-iid data via group-based meta-learning. ACM Transactions on Knowledge Discovery from Data, 17(4), 1-20.

[2] McMahan B, Moore E, Ramage D, et al. Communication-efficient learning of deep networks from decentralized data[C]//Artificial intelligence and statistics. PMLR, 2017: 1273-1282.

[3] Sun J, Li A, Duan L, et al. FedSEA: a semi-asynchronous federated learning framework for extremely heterogeneous devices[C]//Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems. 2022: 106-119.

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

[5] Vahidian S, Morafah M, Wang W, et al. Efficient distribution similarity identification in clustered federated learning via principal angles between client data subspaces[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2023, 37(8): 10043-10052.

[6] Li Z, Shang X, He R, et al. No fear of classifier biases: Neural collapse inspired federated learning with synthetic and fixed classifier[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2023: 5319-5329.

[7] Li T, Sahu A K, Zaheer M, et al. Federated optimization in heterogeneous networks[J]. Proceedings of Machine learning and systems, 2020, 2: 429-450.

[8] Karimireddy S P, Kale S, Mohri M, et al. Scaffold: Stochastic controlled averaging for federated learning[C]//International conference on machine learning. PMLR, 2020: 5132-5143.

[9] Li T, Hu S, Beirami A, et al. Ditto: Fair and robust federated learning through personalization[C]//International conference on machine learning. PMLR, 2021: 6357-6368.

[10] Shin J, Li Y, Liu Y, et al. Fedbalancer: Data and pace control for efficient federated learning on heterogeneous clients[C]//Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services. 2022: 436-449.

[11] Li C, Zeng X, Zhang M, et al. PyramidFL: A fine-grained client selection framework for efficient federated learning[C]//Proceedings of the 28th Annual International Conference on Mobile Computing And Networking. 2022: 158-171.

[12] Yang L, Huang J, Lin W, et al. Personalized federated learning on non-iid data via group-based meta-learning[J]. ACM Transactions on Knowledge Discovery from Data, 2023, 17(4): 1-20.

[13] Tan A Z, Yu H, Cui L, et al. Towards personalized federated learning[J]. IEEE Transactions on Neural Networks and Learning Systems, 2022.

[14] Vettoruzzo A, Bouguelia M R, Rögnvaldsson T. Personalized Federated Learning with Contextual Modulation and Meta-Learning[C]//Proceedings of the 2024 SIAM International Conference on Data Mining (SDM). Society for Industrial and Applied Mathematics, 2024: 842-850.

[15] Fallah A, Mokhtari A, Ozdaglar A. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach[J]. Advances in Neural Information Processing Systems, 2020, 33: 3557-3568.

[16] Li X, Liu S, Zhou Z, et al. EchoPFL: Asynchronous Personalized Federated Learning on Mobile Devices with On-Demand Staleness Control[J]. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 2024, 8(1): 1-22.

[17] Ouyang X, Xie Z, Zhou J, et al. Clusterfl: a similarity-aware federated learning system for human activity recognition[C]//Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services. 2021: 54-66.

[18] Vettoruzzo A, Bouguelia M R, Vanschoren J, et al. Advances and challenges in meta-learning: A technical review[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2024.

[19] Wang K, Mathews R, Kiddon C, et al. Federated evaluation of on-device personalization[J]. arXiv preprint arXiv:1910.10252, 2019.

[20] Zhang L, Shen L, Ding L, et al. Fine-tuning global model via data-free knowledge distillation for non-iid federated learning[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022: 10174-10183.

[21] Ouyang X, Xie Z, Fu H, et al. Harmony: Heterogeneous multi-modal federated learning through disentangled model training[C]//Proceedings of the 21st Annual International Conference on Mobile Systems, Applications and Services. 2023: 530-543.

[22] Smith V, Chiang C K, Sanjabi M, et al. Federated multi-task learning[J]. Advances in neural information processing systems, 2017, 30.

[23] Deng Y, Kamani M M, Mahdavi M. Adaptive personalized federated learning[J]. arXiv preprint arXiv:2003.13461, 2020.

[24] Mansour Y, Mohri M, Ro J, et al. Three approaches for personalization with applications to federated learning[J]. arXiv preprint arXiv:2002.10619, 2020.

[25] Rabanser S, Günnemann S, Lipton Z. Failing loudly: An empirical study of methods for detecting dataset shift[J]. Advances in Neural Information Processing Systems, 2019, 32.

[26] Huang K, Yang B, Gao W. ElasticTrainer: Speeding Up On-Device Training with Runtime Elastic Tensor Selection[C]//Proceedings of the 21st Annual International Conference on Mobile Systems, Applications and Services. 2023: 56-69.

[27] Jothimurugesan E, Hsieh K, Wang J, et al. Federated learning under distributed concept drift[C]//International Conference on Artificial Intelligence and Statistics. PMLR, 2023: 5834-5853.

[28] Zhang W, Zhou Z, Wang Y, et al. DM-PFL: Hitchhiking Generic Federated Learning for Efficient Shift-Robust Personalization[C]//Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. 2023: 3396-3408.

[29] Liu J, Jia J, Che T, et al. Fedasmu: Efficient asynchronous federated learning with dynamic staleness-aware model update[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2024, 38(12): 13900-13908.

[30] You L, Liu S, Wang T, et al. AiFed: An adaptive and integrated mechanism for asynchronous federated data mining[J]. IEEE Transactions on Knowledge and Data Engineering, 2023.

[31] Zhang T, Gao L, Lee S, et al. TimelyFL: Heterogeneity-aware Asynchronous Federated Learning with Adaptive Partial Training[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2023: 5064-5073.

[32] Wu W, He L, Lin W, et al. SAFA: A semi-asynchronous protocol for fast federated learning with low overhead[J]. IEEE Transactions on Computers, 2020, 70(5): 655-668.

[33] Ma Q, Xu Y, Xu H, et al. FedSA: A semi-asynchronous federated learning mechanism in heterogeneous edge computing[J]. IEEE Journal on Selected Areas in Communications, 2021, 39(12): 3654-3672.

[34] Yan G, Wang H, Li J. Seizing critical learning periods in federated learning[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2022, 36(8): 8788-8796.

[35] Yan G, Wang H, Yuan X, et al. Criticalfl: A critical learning periods augmented client selection framework for efficient federated learning[C]//Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. 2023: 2898-2907.

[36] Li T, Hu S, Beirami A, et al. Ditto: Fair and robust federated learning through personalization[C]//International conference on machine learning. PMLR, 2021: 6357-6368.

[37] Qin Z, Yang L, Wang Q, et al. Reliable and interpretable personalized federated learning[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2023: 20422-20431.

[38] Jothimurugesan E, Hsieh K, Wang J, et al. Federated learning under distributed concept drift[C]//International Conference on Artificial Intelligence and Statistics. PMLR, 2023: 5834-5853.

[39] Reisizadeh A, Farnia F, Pedarsani R, et al. Robust federated learning: The case of affine distribution shifts[J]. Advances in Neural Information Processing Systems, 2020, 33: 21554-21565.

[40] Xie C, Koyejo S, Gupta I. Asynchronous federated optimization[J]. arXiv preprint arXiv:1903.03934, 2019.

[41] Park J, Han D J, Choi M, et al. Sageflow: Robust federated learning against both stragglers and adversaries[J]. Advances in neural information processing systems, 2021, 34: 840-851.

[42] Sun J, Li A, Duan L, et al. FedSEA: a semi-asynchronous federated learning framework for extremely heterogeneous devices[C]//Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems. 2022: 106-119.

[43] Zhang L, Shen L, Ding L, et al. Fine-tuning global model via data-free knowledge distillation for non-iid federated learning[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022: 10174-10183.
[44] Wang H, Li Y, Xu W, et al. Dafkd: Domain-aware federated knowledge distillation[C]//Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition. 2023: 20412-20421.
[45] Wang H, Yurochkin M, Sun Y, et al. Federated learning with matched averaging[J]. arXiv preprint arXiv:2002.06440, 2020.
[46] Huang K, Yang B, Gao W. ElasticTrainer: Speeding Up On-Device Training with Runtime Elastic Tensor Selection[C]//Proceedings of the 21st Annual International Conference on Mobile Systems, Applications and Services. 2023: 56-69.
[47] Li C, Zeng X, Zhang M, et al. PyramidFL: A fine-grained client selection framework for efficient federated learning[C]//Proceedings of the 28th Annual International Conference on Mobile Computing And Networking. 2022: 158-171.
[48] Lai F, Zhu X, Madhyastha H V, et al. Oort: Efficient federated learning via guided participant selection[C]//15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21). 2021: 19-35.
[49] Wu W, He L, Lin W, et al. SAFA: A semi-asynchronous protocol for fast federated learning with low overhead[J]. IEEE Transactions on Computers, 2020, 70(5): 655-668.
[50] Sun J, Li A, Duan L, et al. FedSEA: a semi-asynchronous federated learning framework for extremely heterogeneous devices[C]//Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems. 2022: 106-119.
[51] Horvath S, Laskaridis S, Almeida M, et al. Fjord: Fair and accurate federated learning under heterogeneous targets with ordered dropout[J]. Advances in Neural Information Processing Systems, 2021, 34: 12876-12889.
[52] Li D, Wang J. Fedmd: Heterogenous federated learning via model distillation[J]. arXiv preprint arXiv:1910.03581, 2019.
[53] Ghosh A, Chung J, Yin D, et al. An efficient framework for clustered federated learning[J]. Advances in Neural Information Processing Systems, 2020, 33: 19586-19597.
[54] Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network[J]. arXiv preprint arXiv:1503.02531, 2015.
[55] Tan A Z, Yu H, Cui L, et al. Towards personalized federated learning[J]. IEEE Transactions on Neural Networks and Learning Systems, 2022.
[56] Ye M, Fang X, Du B, et al. Heterogeneous federated learning: State-of-the-art and research challenges[J]. ACM Computing Surveys, 2023, 56(3): 1-44.
[57] Zhao Y, Li M, Lai L, et al. Federated learning with non-iid data[J]. arXiv preprint arXiv:1806.00582, 2018.
[58] Chen D, Wang D, Darrell T, et al. Contrastive test-time adaptation[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022: 295-305.
[59] Zhang M, Levine S, Finn C. Memo: Test time robustness via adaptation and augmentation[J]. Advances in neural information processing systems, 2022, 35: 38629-38642.
[60] Nguyen A T, Torr P, Lim S N. Fedsr: A simple and effective domain generalization method for federated learning[J]. Advances in Neural Information Processing Systems, 2022, 35: 38831-38843.
[61] Qu Z, Li X, Duan R, et al. Generalized federated learning via sharpness aware minimization[C]//International conference on machine learning. PMLR, 2022: 18250-18280.
[62] Sanh V, Debut L, Chaumond J, et al. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter[J]. arXiv preprint arXiv:1910.01108, 2019.
[63] Howard A G, Zhu M, Chen B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv preprint arXiv:1704.04861, 2017.
[64] Flower Summit 2023 — Federated Learning Meets the shop floor in Siemens PCB production facilities. (2023). YouTube. Retrieved June 14, 2024, from https://www.youtube.com/watch?v=uxIcS7n5vHg
[65] Jiang L, Lin T. Test-time robust personalization for federated learning[J]. arXiv preprint arXiv:2205.10920, 2022.
[66] Kim H, Kwak Y, Jung M, et al. ProtoFL: Unsupervised Federated Learning via Prototypical Distillation[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2023: 6470-6479.
[67] Li Y, Wang X, An L. Hierarchical clustering-based personalized federated learning for robust and fair human activity recognition[J]. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 2023, 7(1): 1-38.
[68] Canales L, Martínez-Barco P. Emotion detection from text: A survey[C]//Proceedings of the workshop on natural language processing in the 5th information systems research working days (JISIC). 2014: 37-43.
[69] Krizhevsky A, Hinton G. Learning multiple layers of features from tiny images[J]. 2009.
[70] Hsu T M H, Qi H, Brown M. Measuring the effects of non-identical data distribution for federated visual classification[J]. arXiv preprint arXiv:1909.06335, 2019.
[71] Li Y, Su H, Shen X, et al. Dailydialog: A manually labelled multi-turn dialogue dataset[J]. arXiv preprint arXiv:1710.03957, 2017.
[72] Sanh V, Debut L, Chaumond J, et al. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter[J]. arXiv preprint arXiv:1910.01108, 2019.
[73] Wang K, Mathews R, Kiddon C, et al. Federated evaluation of on-device personalization[J]. arXiv preprint arXiv:1910.10252, 2019.
[74] Reiss T, Hoshen Y. Mean-shifted contrastive loss for anomaly detection[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2023, 37(2): 2155-2162.
[75] Reiss T, Cohen N, Bergman L, et al. Panda: Adapting pretrained features for anomaly detection and segmentation[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021: 2806-2814.
[76] Zeng X, Fang B, Shen H, et al. Distream: scaling live video analytics with workload-adaptive distributed edge intelligence[C]//Proceedings of the 18th Conference on Embedded Networked Sensor Systems. 2020: 409-421.
[77] Hard A, Rao K, Mathews R, et al. Federated learning for mobile keyboard prediction[J]. arXiv preprint arXiv:1811.03604, 2018.
[78] Niu C, Wu F, Tang S, et al. Billion-scale federated learning on mobile clients: A submodel design with tunable privacy[C]//Proceedings of the 26th Annual International Conference on Mobile Computing and Networking. 2020: 1-14.
[79] Bakopoulou E, Tillman B, Markopoulou A. Fedpacket: A federated learning approach to mobile packet classification[J]. IEEE Transactions on Mobile Computing, 2021, 21(10): 3609-3628.
[80] Chen D, Wang D, Darrell T, et al. Contrastive test-time adaptation[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022: 295-305.
[81] Taori R, Dave A, Shankar V, et al. Measuring robustness to natural distribution shifts in image classification[J]. Advances in Neural Information Processing Systems, 2020, 33: 18583-18599.
[82] Hsu T M H, Qi H, Brown M. Measuring the effects of non-identical data distribution for federated visual classification[J]. arXiv preprint arXiv:1909.06335, 2019.
[83] Gong B, Xing T, Liu Z, et al. Adaptive clustered federated learning for heterogeneous data in edge computing[J]. Mobile Networks and Applications, 2022, 27(4): 1520-1530.
[84] Gong B, Xing T, Liu Z, et al. Adaptive client clustering for efficient federated learning over non-iid and imbalanced data[J]. IEEE Transactions on Big Data, 2022.
[85] Gong B, Xing T, Liu Z, et al. Towards Hierarchical Clustered Federated Learning with Model Stability on Mobile Devices[J]. IEEE Transactions on Mobile Computing, 2023.
[86] Jiang S, Shuai X, Xing G. ArtFL: Exploiting Data Resolution in Federated Learning for Dynamic Runtime Inference via Multi-Scale Training[C]//2024 23rd ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN). IEEE, 2024: 27-38.
[87] Fang C, Liu S, Zhou Z, et al. AdaShadow: Responsive Test-time Model Adaptation in Non-stationary Mobile Environments[J]. arXiv preprint arXiv:2410.08256, 2024.
[88] Liu S, Luo H, Li X C, et al. AdaKnife: Flexible DNN Offloading for Inference Acceleration on Heterogeneous Mobile Devices[J]. IEEE Transactions on Mobile Computing, 2024.
[89] Liu S, Li X, Zhou Z, et al. AdaEnlight: Energy-aware low-light video stream enhancement on mobile devices[J]. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 2023, 6(4): 1-26.
[90] Liu S, Guo B, Fang C, et al. Enabling resource-efficient aiot system with cross-level optimization: A survey[J]. IEEE Communications Surveys & Tutorials, 2023.
[91] Liu S, Guo B, Ma K, et al. AdaSpring: Context-adaptive and runtime-evolutionary deep model compression for mobile applica-

tions[J]. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 2021, 5(1): 1-22.

[92] Wang H, Guo B, Liu J, et al. Context-aware adaptive surgery: A fast and effective framework for adaptative model partition[J]. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 2021, 5(3): 1-22.

[93] Chu H, Zheng X, Liu L, et al. nnPerf: Demystifying DNN Runtime Inference Latency on Mobile Platforms[C]//Proceedings of the 21st ACM Conference on Embedded Networked Sensor Systems. 2023: 125-137.

[94] Wang Z, Liu K, Hu J, et al. Attrleaks on the edge: Exploiting information leakage from privacy-preserving co-inference[J]. Chinese Journal of Electronics, 2023, 32(1): 1-12.

[95] Zhang T, Fu Y, Zhang J, et al. Deep Guided Attention Network for Joint Denoising and Demosaicing in Real Image[J]. Chinese Journal of Electronics, 2024, 33(1): 303-312.

Xiaochen Li is a Master's student at Northwestern Polytechnical University, China. He received B.E. from Northwest University in 2022. His research interests include resource-efficient mobile deep learning and mobile systems.

Sicong Liu received the PhD degree in school of computer science and technology from Xidian University in 2020. Now she is an associate professor with school of computer science, Northwestern Polytechnical University. Her research interests include mobile computing and resource-efficient mobile deep learning. She has served as the TPC member of MobiSys 2021 and BigCom 2021.

Zimu Zhou is currently an assistant professor in the Department of Data Science, City University of Hong Kong. He obtained his Ph.D. from the Hong Kong University of Science and Technology in 2016 and B.E. from Tsinghua University in 2011. His research interest lies broadly in mobile and ubiquitous computing, with a focus on AIoT. zimuzhou@cityu.edu.hk Zimu Zhou's research is supported by CityU APRC grant (No. 9610633).

Yuan Xu is a Master's student at Northwestern Polytechnical University, China. He received his B.E. from Northwest University in 2023. His interest is in Artificial Intelligence of Things (AIoT).

Bin Guo received the PhD degree in computer science from Keio University, Japan, and then was a postdoc researcher with Institut Telecom SudParis, France. He is a professor with Northwestern Polytechnical University, China. His research interests include ubiquitous computing, mobile crowd sensing, and HCI. He has served as an associate editor of the IEEE Communications Magazine and the IEEE Transactions on Human-Machine-Systems, the guest editor of the ACM Transactions on Intelligent Systems.

Zhiwen Yu received the Ph.D. degree in computer science from Northwestern Polytechnical University, Xi'an, China, in 2005. He is currently a Professor and the Dean of the School of Computer Science, Northwestern Polytechnical University, Xi'an, China. He was an Alexander Von Humboldt Fellow with Mannheim University, Germany, and a Research Fellow with Kyoto University, Kyoto, Japan. His research interests include ubiquitous computing, HCI, and mobile sensing and computing.