AdaCLP: Efficient Federated Learning for Asynchronous Mobile Devices with Temporally Imbalanced Data

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

Abstract—Federated learning (FL) enables mobile devices to collaboratively train deep learning models while maintaining data privacy and minimizing communication overhead. However, traditional FL methods typically assume access to pre-collected datasets, an impractical assumption for real-world mobile devices that continuously collect sensor data streams without retaining them while operating, due to limited memory constraints. Streaming Federated Learning (SFL) partially addresses this limitation by supporting online learning and asynchronous model aggregation directly from live data streams. Yet, a critical challenge in mobile data streams is the *temporal class imbalance*, which may biase the streaming FL process toward early-arriving data classes during the critical learning period (CLP), the initial training phase when the model exhibits the highest plasticity. To address this, we propose AdaCLP, a training scheduler that tracks global model plasticity using a staleness-decayed mechanism specifically designed for the dynamics of real-world mobile environments. AdaCLP dynamically adjusts local training hyperparameters in SFL to prolong the CLP, thereby preserving model plasticity in the face of time-varying, imbalanced data distributions. Furthermore, a CLP-aware dynamic voltage and frequency scaling (DVFS) strategy is integrated to reduce the energy cost of prolonged training, aligning with the tight energy budgets of mobile devices. Experimental results demonstrate that AdaCLP can effectively handle either temporally imbalanced or periodic data, improving accuracy by up to 11% while reducing energy consumption by 69.5% compared to state-of-the-art methods, without modifying the standard FL training pipeline. These demonstrate that AdaCLP is a practical and efficient solution for real-world mobile FL deployment, enabling robust on-device adaptation to evolving data streams.

Index Terms—Streaming Federated Learning, Critical Learning Period, Temporal Class Imbalance, Mobile Devices

1 Introduction

Federated Learning (FL) [2] empowers mobile devices (clients) to collaboratively train deep neural networks (DNNs) under coordination of a server while keeping their data localized. This paradigm has been adopted in various mobile computing applications such as activity recognition [24], [25], [39], personalized recommendations [22], [30], [48], intelligent transportation [28], [33], [36], [49], *etc.*, enhancing data privacy and reducing communication overhead. Traditionally, FL assumes that all devices have *pre-collected*, *static* datasets ready for training. This assumption introduces significant inefficiencies in real-world mobile scenarios, as it requires a lengthy data accumulation phase before training can begin, particularly impractical for mobile devices with limited memory.

For example, consider federated updates of a human activity recognition model in applications like the Google Fit App [69] or Apple Health [70], which may need daily model refreshes. A typical smartwatch can generate around 1.2 GB of accelerometer data per day, yet mobile devices like the Apple Watch Series 4 offer less than 32 GB of total storage, shared across all apps and system processes [51]. Under such constraints, storing several days' worth of sensor data for training is infeasible, exposing a fundamental conflict between traditional FL assumptions and real-world mobile deployment.

Streaming Federated Learning (SFL) [3] emerges as a promising alternative by enabling online training directly on continuous data streams, thereby eliminating the need for extensive storage or long data accumulation periods. Rather than waiting for sufficient data to be gathered, SFL allows immediate local updates as data arrives, making it particularly suited to mobile environments characterized by limited resources and rapid data turnover. A further advantage of SFL lies in its support for asynchronous model aggregation [4], [62], where updates from mobile clients are integrated by the server as they arrive. This is critical in open-world mobile scenarios, where mobile clients' local training times can vary significantly due to heterogeneity in computing resources, network connectivity, and battery levels. Without asynchronous aggregation, FL systems must either wait for straggling devices, introducing latency—or proceed prematurely, sacrificing accuracy, both of which undermine deployment efficiency.

Despite these advantages, the practical deployment of SFL in real-world mobile applications remains challenging. Foremost among them is the issue of *temporal heterogeneity* in mobile data streams. Unlike traditional FL, where large, pre-collected datasets are often curated to be class-balanced, SFL operates on live sensor data, preserving *only a small window* of recent samples for training. Within such limited temporal windows, the data distribution is often highly imbalanced. For example, a user fitness tracking app may receive mostly sleep-related data during the night

**Corresponding Author: Zhiwen Yu (zhiwenyu@nwpu.edu.cn)*
Sicong Liu, Yuan Xu, Xiaochen Li, Wieye Wu, and Bin Guo with the Northwestern Polytechnical University, Xi'an 710100, China. Zimu Zhou, with the City University of Hong Kong, Hong Kong 999077, China. Zhiwen Yu with the Northwestern Polytechnical University, Xi'an 710072 *and Harbin Engineering University, Harbin* 150001, *China.*

and walking or running data during the day (see Fig. 1). Since SFL updates the model incrementally based on the current window, the model may overfit to the most recent dominant class, *e.g.*, learning only to recognize sleep if training happens overnight. This temporal class imbalance undermines generalization and compromises the model's long-term effectiveness across diverse activity types.

Existing methods attempt to mitigate temporal imbalance by rebalancing class distributions through *historical data retrieval* [3], [38], [52], [62] or *knowledge distillation* [5], [13], [19], [46], [65], [66]. However, they require storing samples of old classes or rely on additional public datasets, which contradicts SFL's goal of lightweight, memory-efficient mobile learning. Other method leverage *meta learning* [37], *parameter specialization* [17], or *calibration methods* [67] to improve adaptation to evolving distributions. Yet, they require modifications to training losses or procedures, making them difficult to integrate with existing FL frameworks.

Fundamentally, we *observe* that the *temporal imbalance issue in mobile SFL* is fundamentally tied to the model's critical learning period (CLP) [12], during which the model exhibits high plasticity and rapidly assimilates incoming data. As training continues, this plasticity naturally declines, making the model increasingly resistant to updates from new classes that appear later in the stream. Consequently, if early-arriving classes dominate during the CLP, the model becomes biased toward them, impairing its ability to adapt to later-emerging classes and thus compromising generalization across the full range of target activities.

Motivated by this observation, we tackle the temporal class imbalance problem in SFL from an *orthogonal* perspective. Instead of enlarging data buffers or modofying training losses, we present to extend the model's critical learning period (CLP), *i.e.*, a phase marked by high plasticity and sensitivity to diverse data. The key idea is inspired by "stay hungry, stay foolish". Also, the feasibility of extending the CLP rests on two well-known facts: *i)* the CLP is both measurable and tunable through training hyperparameters like learning rate and batch size. *ii)* its duration is naturally limited by the recurring patterns of classes in the data stream. For example, extending the CLP to cover one full day can capture diverse daily activities without indefinite extension (see Fig. 1). However, extending the CLP in mobile SFL is non-trivial.

- ***Challenge #1: CLP measurement under asynchronous SFL***. Asynchronous SFL [64] improves latency by aggregating models as they arrive, but this desynchronization complicates CLP estimation. Centralized or synchronous SFL computes CLP via the Fisher Information Matrix (FIM) [1], [12]. While asynchronous settings involve local models at varying stages, naively averaging their FIMs yields inaccurate estimates due to staleness.
- ***Challenge #2: Managing energy cost of prolonged CLP***. Extending the CLP requires tuning hyperparameters that increase local computation and communication, raising energy consumption, a major constraint for mobile devices. Although prior work explores CLP for improving learning [12], [32], [32], few consider computation overhead [32], and none offer energy-efficient CLP extension strategies for on-device SFL.

To this end, we introduce AdaCLP, a training scheduler to enhance the efficiency of mobile SFL by dynamically extending CLP. AdaCLP features a federated CLP regulator that monitors the plasticity of the global model and extends the CLPs of local training. It adopts a staleness-aware aggregation of local Fisher Information Matrices (FIMs), ensuring accurate plasticity measurement in asynchronous updates. These measurements are then propagated to mobile clients, who then adjust training hyperparameters, such as learning rate, batch size, and dropout, to prolong local CLPs. To manage the energy cost of extended CLPs, AdaCLP uses the dynamic voltage frequency scaling (DVFS) method available in modern mobile devices [8], [9], [57], in a CLP-aware manner. During the CLP, it lowers CPU/GPU frequencies to to conserve energy and slow down wall-clock training, effectively lengthening the CLP duration. After the CLP ends, AdaCLP gradually restores frequencies to accelerate convergence. Our main contributions are summarized as follows.

- To our knowledge, this is the first to incorporate adaptive CLP control into SFL to address temporal class imbalance arising from user-driven data streams and limited mobile data buffer. It requires no modifications to existing applications or learning procedures.
- We develop AdaCLP, a plasticity-aware learning scheduler to enhance the accuracy of SFL on mobile devices. AdaCLP ensures accurate CLP estimation under asynchronous updates and manages energy cost from prolonged training, making it practical for deployment on heterogeneous, battery-constrained mobile devices.
- Extensive experiments across diverse mobile tasks, platforms, and scenarios show that AdaCLP consistently outperforms classical and continual FL (both synchronous and asynchronous), achieving up to 11% accuracy gain and 69.5% energy reduction. Moreover, AdaCLP can be combined with existing methods for further improvements (see Sec. 5.4.4).

2 Overview

This section presents an overview of AdaCLP, an CLP-adaptive training scheduler for streaming federated learning on mobile devices.

2.1 Motivation

As mentioned in Sec. 1, streaming federated learning (SFL) [3] enables online, asynchronous model training on data streams, thus reducing storage and latency for data collection on mobile devices. However, SFL often suffers from degraded accuracy due to *temporal class imbalance* inherent in user-driven data streams, limited mobile buffers, and the streaming learning paradigm itself. Unlike traditional FL, which relies on large, pre-collected, and often class-balanced datasets, SFL trains on live sensor data while preserving *only a small, recent window of samples*. Within these limited temporal windows, class distributions tend to be highly imbalanced. Moreover, early-arriving classes dominate the initial training phase, causing the model to overfit to these classes and hindering its ability to generalize to classes appearing later.

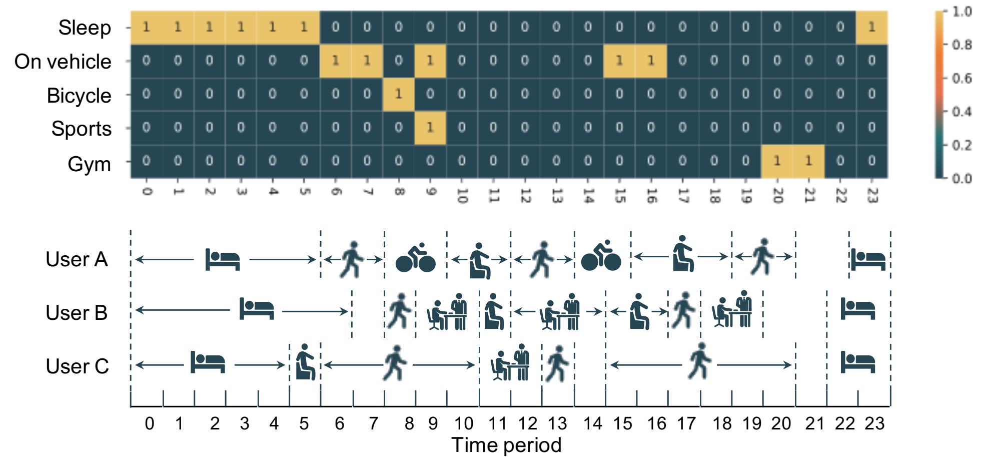


Fig. 1. Illustration of *temporal class imbalance* of data streams collected by different mobile users for human activity recognition.

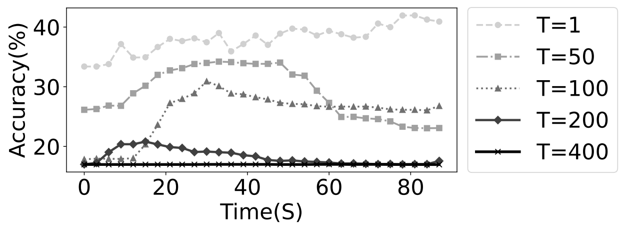


Fig. 2. Impact of sequential learning on accuracy when trained on continuously varying single-class data (*e.g.*, walking, hopping, phone calls, waving, and typing) over a period T ($T = 1, 50, 100, 200, 400$) across 120 mobile clients.

To illustrate this issue, we conduct an experiment with 120 mobile clients using the SFL framework, where the server enforces sequential class learning, *i.e.*, each client trains on a single class for T rounds before transitioning to the next class. We use five classes (*e.g.*, walking, hopping, phone calls, waving, and typing) from the HARBox dataset [39] and vary T across 1, 50, 100, 200, and 400 rounds. As shown in Fig. 2, increasing T, *i.e.*, prolonging the period a class is exclusively trained on, leads to a clear decline in the model's overall accuracy across all classes. This demonstrates how temporal class imbalance, amplified by mobile SFL's small buffer windows and streaming nature, severely degrades performance and underscores the need for training mechanisms to mitigate this imbalance.

2.2 Design Rationales

In mobile environments, the arrival patterns of data classes are dynamic and inherently nonstatistical, they are influenced by diverse user behaviors, objectives, and dynamic environmental factors. This *unpredictability* leads to agnostic temporal class imbalance, where certain classes may dominate the data stream during specific periods, skewing the model's learning process. Our key *insight* is that this issue is *fundamentally tied to the critical learning period (CLP) in SFL*, a phase where the model exhibits high plasticity, meaning it rapidly absorbs and adapts to incoming data. However, as training progresses, the model's plasticity diminishes, making it less receptive to new class distributions appearing later in the stream. This insight *suggests an orthogonal approach*, *i.e.*, instead of controlling the uncontrollable data arrival patterns or extending the data buffer, we can prolong the CLP in SFL, keeping the model in its high-plasticity state for an extended duration. This allows the model to incorporate a broader range of classes before stabilization, mitigating the overfitting bias toward early-arriving classes. This approach is particularly feasible for two reasons:

- We can *identify* whether a DNN is within its CLP by measuring the Fisher information of its model parameters [1], [12], [32] and *prolong* the CLP by adjusting the learning rate, batch size, and dropout rate, *etc.* during training [12], [43]. This process allows for fine-tuned control over the model's capacity to absorb new classes.
- We can harness the statistical periodicity of class appearances in mobile applications to *bound the maximum CLP extension*. Mobile apps, such as those for human activity recognition, experience classes of data in a cyclical or predictable pattern. By extending the CLP just a few times beyond its typical duration, measured in rounds, we can mitigate the problem of temporal class imbalance, effectively handling situations where some classes appear more frequently or at certain times. This is crucial for ensuring that the model can generalize well, even in the face of imbalanced data.

2.3 System Architecture

AdaCLP transforms the above rationales into a practical FL system tailored for asynchronous, resource-limited, and data-imbalanced mobile devices. It consists of two major modules, as illustrated in Fig. 3, each designed to address significant challenges:

- *Federated Critical Learning Period Regulator (Sec. 3).* To address Challenge #1, this module accurately *measures* and *controls* the CLP of the global model with *asynchronous* mobile clients. Traditional CLP was only measured in centralized training [11], [12] and synchronous FL [1]. We adapt CLP measurements to asynchronous FL with *dedicated staleness control* at the server. Furthermore, we jointly configure key hyperparameters, such as learning rate, batch size, and dropout rate, at each mobile client to actively prolong the CLP, ensuring better plasticity and generalization in the model.

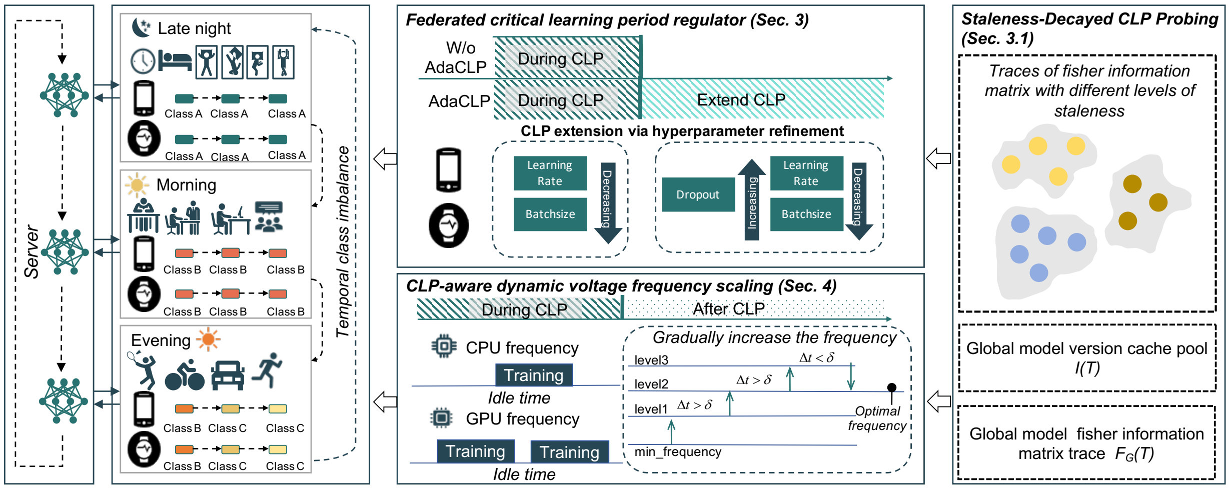

Server
Late night
Class A
Morning
Class B
Evening
Class C
Temporal class imbalance
Federated critical learning period regulator (Sec. 3)
W/o AdaCLP
During CLP
AdaCLP
Extend CLP
CLP extension via hyperparameter refinement
Learning Rate
Batchsize
Decreasing
Dropout
Increasing
CLP-aware dynamic voltage frequency scaling (Sec. 4)
After CLP
CPU frequency
Training
Idle time
GPU frequency
Gradually increase the frequency
level3
level2
level1
min_frequency
Optimal frequency
Staleness-Decayed CLP Probing (Sec. 3.1)
Traces of fisher information matrix with different levels of staleness
Global model version cache pool I(T)
Global model fisher information matrix trace F_G(T)

Fig. 3. Overview of AdaCLP for streaming federated learning with asynchronous mobile devices and temporal imbalanced mobile data.

- *CLP-Aware Dynamic Voltage Frequency Scaling (Sec. 4).* To resolve Challenge #2, this module minimizes the additional energy consumption at mobile clients (battery-powereed mobile devices) due to extended CLP via dynamic voltage frequency scaling (DVFS). We utilize DVFS in a CLP-aware manner, the built-in energy-saving technique in modern mobile devices to adjust the power consumption during and after CLP. By lowering processor frequencies during CLP, we conserve energy, implicitly extending CLP in wall-clock time, and by increasing frequencies after CLP, we accelerate local training. We further enhance the FL system by developing frequency scaling strategies for both CPU+GPU and CPU-only mobile devices.

3 Federated Critical Learning Period Regulator

As mentioned above, the arrival patterns of mobile data classes are imbalanced and inherently uncontrollable, presenting a significant challenge in streaming federated learning. To better handle the dynamic nature of real-world data streams, where the arrival of new classes may be sporadic and imbalanced, we aim to ensure more robust learning and improved performance over time. Rather than trying to control these unpredictable patterns, we propose extending the Critical Learning Period (CLP), keeping the model in a *high-plasticity* state for a longer duration. This allows it to adapt to late-arriving new classes without prematurely *stabilizing*, thereby mitigating the negative effects of temporal class imbalance in sequential data.

This section demonstrates how AdaCLP addresses this challenge by measuring and extending the CLP within the context of asynchronous and streaming federated learning. By adapting the CLP extension strategy to work in asynchronous settings, we ensure that the model remains sufficiently plastic to accommodate late-arriving classes, regardless of their timing or frequency.

3.1 Staleness-Decayed CLP Probing

We first explain how to measure the critical learning period (CLP) in streaming federated learning with asynchronous mobile clients. Rather than a fixed interval, we treat the CLP as a tunable metric of high plasticity, quantified by the trace of the Fisher Information Matrix (FIM). This measurement offers a principled and generalizable signal under distributed, label-sparse, and temporally evolving updates. And we adapt existing CLP measurement techniques to asynchronous SFL settings characterized by heterogeneous mobile devices and stale client updates.

3.1.1 CLP Premier

The critical learning period (CLP) [12] of a DNN is associated with the empirical observation that the DNN exhibits high plasticity in the early training epochs and its plasticity declines drastically over time. This non-trivial phenomenon has been adopted to explain the problem of *overfitting* to *early-arriving classes* in sequential learning. Notably, excluding key classes during the CLP can result in irreversible accuracy degradation.

The CLP represents the duration (in epochs) when the DNN maintains high *plasticity*, and can be measured by the ***Fisher Information matrix*** (FIM) trace F of model weights w in *centralized* training [11], [12].

$$F = \mathbb{E}_{x\sim X,\hat{y}\sim p_w(y|x)}\left[||g(x,\hat{y})||^2\right] \tag{1}$$

where $g(x,\hat{y})$ is the gradient of the loss computed for a sample (x,y) via model $p_w(y|x)$ parameterized by w, and $||\cdot||^2$ is the squared norm. Studies show that F increases rapidly during initial training, and drops as training continues. A predefined threshold is often utilized to mark the *turning point*, *i.e.*, the end of the CLP.

3.1.2 Limitations of Prior CLP Measurement Methods

In federated learning, the CLP can be calculated as the weighted average of the Fisher Information Matrix traces of local model weights [1]. However, it only applies to *synchronous* clients because, in synchronous FL, the weighted average of the Fisher information matrix trace of local model weights inherently represents the training effect of the previous global model across the selected clients' data.

In contrast, *asynchronous* FL introduces uncertainty in the global model version received by each client, making

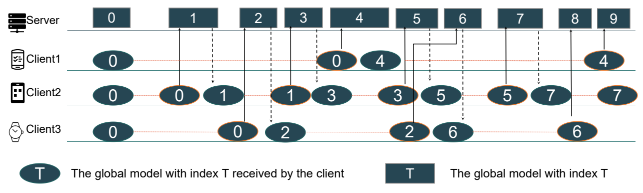

Server
Client1
Client2
Client3
0
1
2
3
4
5
6
7
8
9
The global model with index T received by the client
The global model with index T

Fig. 4. Illustration of *asynchronous* aggregation of models uploaded by diverse fast and slow mobile devices in streaming federated learning.

it challenging to compute the Fisher information matrix trace average based on a consistent global model. In real-world asynchronous scenarios with heterogenous (*e.g.*, fast and slow) mobile devices, the local models are trained from different versions of the global model. This discrepancy, often referred to as ***staleness***, arises because slower devices upload updates based on outdated global models, while faster devices contribute more frequently.

Naively aggregating the Fisher Information Matrix traces derived from these asynchronous local models can lead to *inaccurate CLP measurements*. Fig. 4 illustrates an example of the inconsistency in asynchronous aggregation. In the first 6 rounds, the local models uploaded to the server are from client [2, 3, 2, 1, 2, 3], but the corresponding global model versions (circled in red) are [0, 0, 1, 0, 3, 2]. The inconsistency in the start point (*i.e.*, global model versions due to staleness) of local FIM traces causes errors in estimating the CLP, as the FIM traces are influenced by the specific model initialization and the data encountered during training.

3.1.3 Measuring CLP with Asynchronous Mobile Clients

To address this *unique challenge* in real-world *asynchronous* mobile scenarios, we propose a *staleness-aware CLP measurement* method for SFL. Inspired by weight caching and decay techniques commonly used for effective model aggregation in asynchronous federated learning, our approach adapts these methods to ensure accurate CLP assessment in scenarios with asynchronous clients.

- We cache and track the versions of global models. Specifically, each global model on the server is assigned a version index $T = 1, 2, ...$ which increments after each new aggregation. Mobile clients receive the global model with a version index, allowing them to trace back to the corresponding global model when uploading their local models for CLP calculation.
- Then, we compute the CLP on a sliding window basis, and penalize the outdated local FIM traces via an exponential decay. Let $F_{received}(T)$ be the local FIM trace received by the server in the T-th round, where $F_{received}(T)$ is calculated by each client as Equ.(1). The Fisher Information Matrix trace of the global model is then estimated as:

$$F_G(T) = \frac{1}{m}\sum_{j=0}^{m-1} e^{-\lambda[(T-j)-I(T-j)]} F_{received}(T-j) \tag{2}$$

where m is the window size, and $I(T)$ represents the original version index of the local model aggregated in the T-th round, which is trackable due to our caching mechanism above. λ is a hyperparameter to penalize the staleness of local FIM traces.

We illustrate the calculation process of the global FIM trace in Fig. 5. As shown in Fig. 5a, we present the traditional calculation method used in synchronous FL. In each global round, the server collects the FIM trace uploaded by all clients in the current round and performs an average aggregation. Since all clients in the same round share the same model index (having received the same global model), there is no interference caused by staleness, allowing for precise CLP measurement.

However, directly applying this method to asynchronous settings is still non-trivial. As shown in Fig. 5b, simply extending the *computation window* across multiple rounds and averaging the FIM trace within the window results in *substantial errors* due to staleness. For instance, when computing with models from rounds [2, 3, 4], the model received in round 4 may have a version index of 0, which deviates significantly from the current round. Averaging these traces would lead to bias, undermining the accuracy of CLP measurement.

Our proposed asynchronous FIM measurement method is shown in Fig. 5c to address these challenges. Leveraging the cached global model version indices, we apply decay penalties proportional to the staleness (*e.g.*, the difference between the current round and the version index of the client model). This mitigates the impact of staleness on CLP measurement. For example, when computing the FIM trace for rounds [2, 3, 4], the global version indices are [0, 1, 0]. Correspondingly, we apply decay weights of $[e^{-2\lambda}, e^{-2\lambda}, e^{-4\lambda}]$ to each trace, effectively balancing the staleness impact and enabling accurate global FIM trace measurement in asynchronous settings.

Finally, we determine whether the global model is in the CLP based on the following criterion.

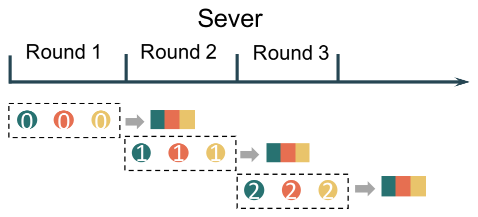
$$\text{CLP} = \begin{cases} \text{True} & \text{if } \frac{F_G(T)-F_G(T-1)}{F_G(T-1)} \geq \sigma \\ \text{False} & \text{otherwise} \end{cases} \tag{3}$$

In essence, the global model is considered to be within the CLP if the growth rate of the global Fisher Information Matrix (FIM) trace exceeds a predefined threshold σ. This approach ensures that significant changes in the FIM trace are appropriately recognized as indicators of high plasticity.

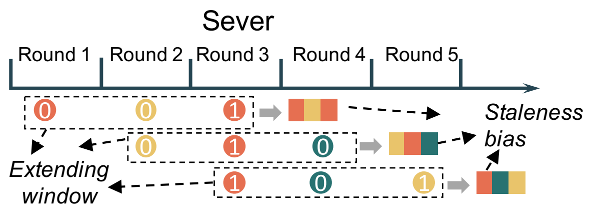
3.2 CLP Extension via Hyperparameter Refinement

As discussed in Sec. 1, extending the CLP during federated learning (FL) enables the model to learn from a broader range of classes while in its high-plasticity state. This approach is particularly effective for handling heterogeneous and periodic data streams. Since the global model's CLP is derived from the aggregated weights of local models, we propose extending the CLP by carefully controlling key *hyperparameters* during *local training*. Specifically, we achieve this by: *i)* reducing the learning rate η, *ii)* decreasing the batch size $\mathcal{B}$, and *iii)* introducing dropout operations $\mathcal{D}$. The details of these strategies are elaborated below:

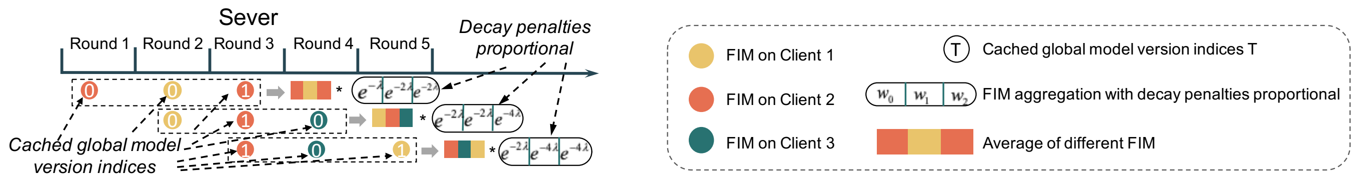
$$\eta = \eta_0 \cdot (\frac{1}{2})^{ln(F_G(T))} = \eta_0 \cdot F_G(T)^{-\ln 2} \tag{4}$$



(a) Traditional method in *synchronous* settings



(b) Traditional method in *asynchronous* settings with extended window


Sever
Round 1
Round 2
Round 3
Round 4
Round 5
Decay penalties proportional
Cached global model version indices
FIM on Client 1
FIM on Client 2
FIM on Client 3
Cached global model version indices T
FIM aggregation with decay penalties proportional
Average of different FIM

(c) The proposed asynchronous method in AdaCLP

Fig. 5. Illustration of the traditional and proposed global FIM trace calculation methods under a-/synchronous settings.

$$\mathcal{B} = 1 + \mathcal{B}_0 \cdot \frac{1}{1 + ln(F_G(T))} \tag{5}$$

$$\mathcal{D} = \mathcal{D}_0 \cdot (1 - \sigma(\beta \cdot (F_G(T) - F_G(T-1)))) \tag{6}$$

where η_0, $\mathcal{B}_0$, and $\mathcal{D}_0$ are the initial learning rate, batch size, and dropout probability, and β is an adjustable hyperparameter, $\sigma(\cdot)$ is the sigmoid function.

In traditional training with non-streaming data, a larger learning rate or smaller batch size typically enables the model to explore a broader parameter space. In continual training with streaming data, it is more effective to gradually reducing the learning rate and batch size [12], [43]. We adopt the scheduling methods proposed in continual learning while dynamically adjusting these parameters based on the current Fisher Information Matrix (FIM) measurement $F_G(T)$. Furthermore, dropout [53] enhances model plasticity by randomly removing trained neurons. To extend the CLP, we dynamically (controlled by α) adjust the dropout rate based on changes in successive FIM measurements, enabling the model to re-enter the CLP.

We provide theoretical insights into the design motivations of our three adaptive scheduling strategies. First, in Equ.(4), we introduce a Fisher-aware decay function, where the learning rate η adaptively responds to the Fisher information trace $F_G(T)$, which reflects the model's confidence and loss landscape curvature. As $F_G(T)$ increases, indicating reduced plasticity, a smaller learning rate mitigates catastrophic forgetting while enabling gradual updates. The power-law form ensures a smoother decay than exponential schemes, effectively prolonging the Critical Learning Period (CLP). Second, in Equ.(5), we reduce the batch size as $F_G(T)$ increases, injecting controlled gradient noise during stable phases to extend the CLP. This design leverages the theoretical observation that smaller batches introduce beneficial stochasticity, helping escape sharp minima and maintain generalization. Our batch-size scheduling is bounded, smooth, and progressively enhances noise in later stages while avoiding numerical instability. Finally, in Equ.(6), we dynamically adjust the dropout rate based on changes in $F_G(T)$. As the model transitions from the CLP to a non-CLP state, indicated by a gradual decrease in $F_G(T) - F_G(T-1)$, the increased dropout strength helps enhance the plasticity of the model structure. Meanwhile, the dynamic dropout mechanism periodically perturbs the network, allowing it to re-enter a plastic state and avoid getting trapped in locally stable states, thereby improving adaptability and continual learning performance.

We conduct a preliminary validation under asynchronous federated learning, and Fig. 6 demonstrates the impact of extending the CLP through hyperparameter tuning. The figure presents the asynchronous CLP curve under three different conditions: without adjustment, adjusting learning rate and batch size, and adjusting dropout rate. Initially, without hyperparameter adjustment, the CLP interval spans rounds [0, 2662]. After applying dynamic learning rate and batch size adjustments, the CLP curve, instead of rapidly declining after reaching the first peak, gradually rises, extending the CLP. Furthermore, with dynamic dropout adjustment, the initial CLP interval is extended from [0, 2662] to [0, 2993], and the model re-enters the CLP for a period after round 5000.

In practical mobile scenarios, we divide the entire FL training process into multiple *segments* and perform the hyperparameter refinement above for each segment. This is clever because, instead of exploiting the natural periodicity of incoming classes in mobile data streams, which would be impractical, we attempt to extend the CLP. By focusing on manageable segments, we ensure that the model adapts efficiently to the continuously changing data distribution and strike a balance between maintaining the model's plasticity and ensuring practical feasibility, allowing the system to handle the dynamic nature of real-world data streams effectively. To achieve this, we *empirically define* a maximum round R_{end} based on the degree of temporal class imbalance, and partition the training process into segments. At the start of each segment, we reset the three hyperparameters η_0, $\mathcal{B}_0$, and $\mathcal{D}_0$. Within each segment R_{end}, we dynamically adjust the learning rate hyperparameters η and

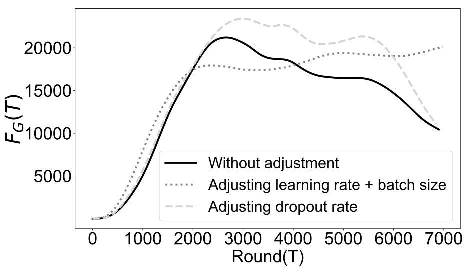


Fig. 6. Performance comparison of adjusting learning rate, batch size, and dropout rates on the critical learning period (CLP).

batch size $\mathcal{B}$ as defined in Equ.(4) and Equ.(5) to extend the CLP, effectively prolonging the CLP. If the model is detected to have left the CLP, we further dynamically increase the dropout rate as Equ.(6)) to help the model to *re-enter* the CLP for high-plasticity state. After R_{end}, we stop tracking the CLP and hyperparameter adjustments, resuming standard asynchronous federated learning to accelerate convergence. This design balances the need for precise CLP management during critical periods with the efficiency required for real-world ubiquitous applications.

4 CLP-Aware Dynamic Voltage Frequency Scaling

Extending the CLP improves model adaptability to imbalanced data streams but also increases energy consumption, a critical concern for battery-powered mobile devices. To address this trade-off, AdaCLP integrates CLP-aware scheduling with system-level energy optimization via Dynamic Voltage and Frequency Scaling (DVFS). Specifically, it introduces a lightweight, FIM-guided DVFS controller that lowers CPU/GPU frequencies during high-plasticity CLP phases to reduce energy usage while preserving model receptiveness to diverse classes. After the CLP ends, frequencies are gradually increased to accelerate convergence during the stable phase. This hybrid coordination between learning dynamics and hardware scheduling allows AdaCLP to retain the benefits of extended CLP while significantly reducing energy overhead. To our knowledge, this is the first FL framework that aligns training plasticity with adaptive DVFS in asynchronous mobile SFL settings. We detail the design below.

4.1 Device-specific Frequency Scaling Principles

Dynamic Voltage and Frequency Scaling (DVFS) [8], [9], [57] is a key technique in mobile computing, that enables mobile processors to adjust their operating frequency dynamically in response to workload demands. Building upon DVFS, we propose a novel frequency scaling strategy specifically designed for the extended Critical Learning Period (CLP) in federated learning, leveraging the unique characteristics of mobile devices. Our strategy integrates *low* processor frequencies *within* the CLP and *high* frequencies *after* the CLP, with the following principles:

- *Extend CLP Measured in Wall-Clock Time*. The hyper-parameter refinement strategies introduced in Sec. 3.2 extend the CLP duration measured in *training rounds*. By dynamically reducing the processor operating frequency during the CLP, each training round consumes more wall-clock time. This extends the CLP in real-time, allowing the model to observe a *wider diversity of data classes* under the same class arrival rate in data streams. Such an approach mitigates temporal class imbalance, a common challenge in asynchronous federated learning on mobile devices. Additionally, the reduced frequency compensates for the increased *energy consumption* associated with the prolonged CLP in training rounds, aligning well with the dynamic energy constraints of mobile devices.
- *Accelerate Post-CLP Training Measured in Wall-Clock Time*. After the CLP, we implement a strategy to *gradually increase* the processor operating frequency, accelerating the local training process while balancing energy efficiency. This approach ensures that training remains effective during the post-CLP phase, where the model transitions to standard federated learning updates. Unlike naive high-frequency settings, which risk unnecessary energy consumption, our gradual scaling strategy aligns with findings from empirical studies, such as [44], which highlight that optimal model training performance often occurs below maximum power levels. By carefully tuning the operating frequency based on workload demands, we achieve faster convergence in wall-clock time without imposing excessive energy costs on resource-constrained mobile devices. This adaptive frequency adjustment not only accelerates post-CLP training but also enhances the practicality of deploying federated learning on heterogeneous mobile platforms, where energy efficiency and real-time responsiveness are critical.

4.2 CLP-aware Frequency Scaling Strategies

Building on the principles outlined in Sec. 4.1, we design operating frequency adjustment strategies tailored for mobile devices. These strategies aim to optimize energy efficiency and training performance during FL. Fig. 7 illustrates the workflow of CLP-aware frequency scaling strategies. We will elaborate on our proposed approach in the following subsections.

4.2.1 Scaling Strategy for CPU-Only Mobile Devices

Assume N increasing CPU operating frequencies $\{f_n\}$, where $n = 0, 1, \dots, N-1$. We configure the CPU frequency in two phases.

- ***Within CLP***: During the critical learning period, we configure the CPU to operate at the lowest available frequency, f_0. This minimizes energy consumption and extends the CLP measured in wall-clock time, allowing the model to process more diverse data classes and address temporal class imbalance effectively.
- ***After CLP***: Once the model exits the CLP, we gradually increase the CPU frequency to accelerate local training. Let the CPU frequency in the $(n-1)$-th training round after the CLP be f_{n-1}. In the n-th round, we *incrementally raise* the frequency to the next available level, f_n, provided the reduction in training time $\Delta t = t_{\text{avg}}(f_{n-1}) - t_{\text{avg}}(f_n)$ exceeds a predefined

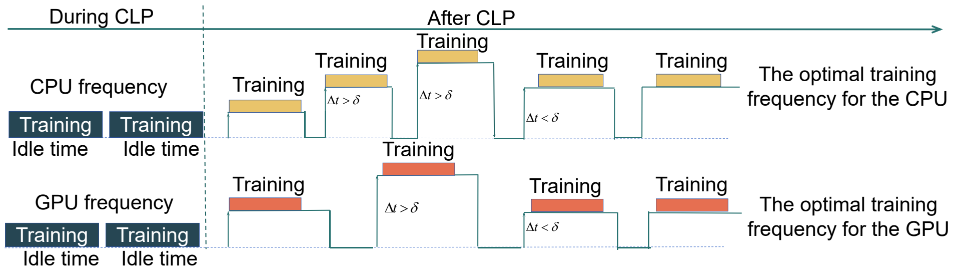


Fig. 7. Illustration of CLP-aware dynamic voltage frequency scaling scheme in mobile devices.

threshold. Here, $t_{\text{avg}}(f)$ represents the average local model training time at frequency f, which is measured by dividing the training time by the number of training data over the recent training iterations. If the increase in frequency no longer yields significant acceleration (*i.e.*, Δt falls below the threshold), we maintain the current frequency to avoid unnecessary energy cost.

The proposed CLP-aware adaptive scaling strategy balances energy efficiency and computational performance, ensuring that battery-powered mobile devices can sustain prolonged CLP in federated learning processes without compromising responsiveness or battery life.

4.2.2 Scaling Strategy for CPU+GPU Mobile Devices

For mobile devices equipped with both CPUs and GPUs, we focus on dynamically scaling the GPU frequency. We follow the principles outlined in Sec. 4.2.1, replacing CPU scaling with GPU scaling, to optimize energy efficiency and fully exploit the GPU's computational power for deep neural network training.

Discussion. As introduced above, AdaCLP dynamically adjusts processor frequency based on the model's position within the CLP. During the CLP, it lowers the frequency to extend wall-clock time, allowing the model to encounter a broader range of data classes, enhancing early-stage class diversity at the cost of fewer local training rounds. This introduces a trade-off between class diversity and training depth, *i.e.*, early exposure to diverse classes improves generalization, while later rounds refine accuracy. As we will show Sec. 5.4.6,by prioritizing class diversity during the plastic phase and emphasizing convergence after CLP, AdaCLP's dynamic frequency scheduling achieves a balanced optimization of model performance and energy efficiency.

5 Experiment

5.1 Experiment Setup

Implementation. We implement AdaCLP using Python 3.11.7 and PyTorch 2.2.2 for the server and mobile clients, respectively. The server is equipped with two RTX 3080 GPUs and 29GB RAM. We use 20 mobile and embedded devices across 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). To simulate heterogeneous computing environments, we configure 120 clients with varying capabilities, distributed as 20% C_1, 20% C_2, 20% C_3, and 40% C_5 by default.

Datasets and Models. We conduct experiments on three datasets for Human Activity Recognition (HAR): CAPTURE-24 [58], HARBox [39], HHAR [47]. Tab. 1 summarizes the dataset configurations. HARBox tracks the daily activities of 121 users (ages 17-55) across 5 activity categories. HHAR involves 9 users performing 6 activities on 12 devices. CAPTURE-24 provides full-day data from 150 users across 11 activity categories. We use two representative models on them: an LSTM (2.1MB) with 900 input features, 128 hidden units, and 1 layer for HarBox, HHAR, and CAPTURE-24 datasets and a Transformer (243MB) with 10 layers and 8 attention heads on the CAPTURE-24.

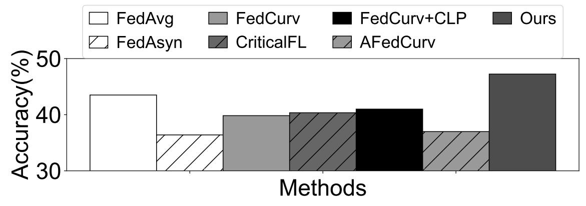
Temporal Class Imbalance Configuration. We simulate the *imbalanced* and *periodic* arrival of data classes in human activity recognition applications in two settings: *extreme* and *non-extreme*.

- ***Extreme setting***: Each client trains only *one* class in each round. Specifically, the server sequentially selects one class C_i from all classes and enforces all clients to train the local model on data of class C_i for a period of T_i before switching to the next class. If a client does not have data of class C_i, it randomly picks another class from its local dataset. The duration of T_i follows a periodic scheme, *i.e.*, Gaussian distribution $T_i \sim N(\mu, \sigma)$, where we set $\mu = 400$ and $\sigma = 50$ to introduce randomness in the switching time.
- ***Non-extreme setting***: Each client trains *multiple* classes in each round. Specifically, the server selects a subset of classes $\{C_i\}$ for all clients to train in each period T_i following the Dirichlet distribution. We vary the parameter β in the Dirichlet distribution to 0.1, 0.2, 0.3, 0.4, and 0.5, to simulate five levels of temporal class imbalance. Other settings are the same as the extreme case.

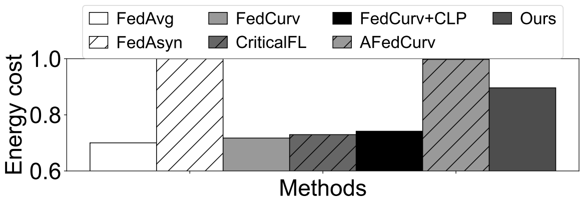
Performance metrics. We evaluate performance using *six* key metrics: accuracy, training time, latency, idle time, and energy consumption. *Accuracy* is calculated as the average accuracy over a period, covering all observed classes. *Latency* is measured as the average time per training round, indicating system responsiveness. *Training time* is determined by multiplying the number of training rounds by the average duration of each round. *Idle time* is obtained by subtracting the training time from the total runtime. For energy consumption, we simulate the total energy usage for a single client: *Energy consumption* is estimated based on the energy consumed during both the training and idle phases. *Total energy consumption* is computed as the sum of energy used in both training and idle periods.

TABLE 1
Overview of datasets.

Dataset	Users	Number of classes	Number of Data Records
HarBox	121	5(Walking, Hopping, Phone calls, Waving and Typing)	32,935
HHAR	9	6(Biking, Sitting, Standing, Walking, Stair Up and Stair down)	69,941
CAPTURE-24	150	11(Sleep, Sitstand+lowactivity, Sitstand+activity, Walking, Vehicle, Sitting, Walking+activity, Bicycling, Standing, Sports, Gym)	934,762



(a) Accuracy, 9 hours



(b) Energy cost, 9 hours

Fig. 8. Comparison of (a) accuracy vs. (b) energy cost between AdaCLP and other baselines in extreme setups over 9 hours.

Baselines. We adopt six representative federated learning (FL) algorithms with mobile devices as performance comparison baselines, reflecting diverse dimensions of FL scenarios. Standard FL methods, including synchronous (FedAvg) and asynchronous (FedAsyn), serve as benchmarks for accuracy and convergence in traditional settings. Federated Continual Learning addresses evolving data streams, pushing the boundaries of continuous adaptation. CLP-based FL emphasizes high accuracy by prioritizing key updates during synchronous training. Hybrid Approaches explore combined constraints, we introduce Synchronous Federated Continual Learning + CLP optimizing temporal performance, while Asynchronous Federated Continual Learning enhances scalability in dynamic environments. They are configured as follows:

- **Classical FL**: The server waits for all clients in each round, setting it as the accuracy baseline because it has the most comprehensive knowledge from all clients. It also sets a high communication cost threshold due to its infrequent communication frequency.
  - **FedAvg** [2] The server calculates the average of all clients' weights and broadcasts the updated global model back to all clients.
  - **FedAsync** [64]: It aggregates the model and distributes updates to clients in an asynchronous manner.
- **Federated Continual Learning (FedCurv)**: **FedCurv** [67] is inspired by Elastic Weight Consolidation (EWC) [56], which calculates the Fisher information matrix of local models and uses it to reduce weight divergence during local training. We use synchronous aggregation here.
- **CLP-based Federated Learning**: **CriticalFL** [32] is based on the CLP metric under synchronous aggregation, ensuring that the global model learns enough category knowledge within the CLP.
- **Hybrid Methods**:
  - **Synchronous Federated Continual Learning + CLP(FedCurv+CLP)**: We combine **FedCurv** [67] with CLP to form a hybrid method.
  - **Asynchronous Federated Continual Learning(AFedCurv**: We replace the synchronous **FedCurv** [67] with asynchronous aggregation.
- **Ours/AdaCLP(FedAsync + CLP)**: Built upon the classical asynchronous federated framework, our approach integrates the Federated CLP Regulator with CLP-based dynamic voltage and frequency scaling (DVFS)

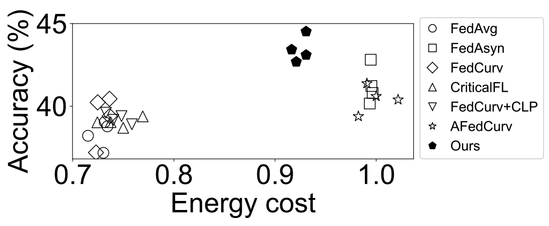
5.2 Performance Comparison

We compare AdaCLP with six baselines, *i.e.*, FedAvg [2], FedAsync [64], FedCurv [67], CriticalFL [32], AFedCurv, FedCurv+CLP on three datasets, *i.e.*, HarBox, HHAR, and CAPTURE-24. We evaluate their overall performance in terms of accuracy and energy cost in both extreme and non-extreme settings.

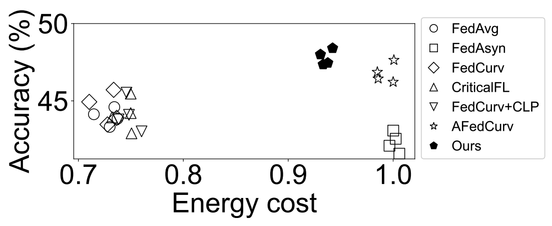
5.2.1 Performance Comparison in Extreme Setups

We first conduct experiments in the extreme setting as Sec. 5.1, using a subset of classes C_i consisting of a single class, with $T = 400$ as the period before switching to the next class. Results are measured over 9-hour periods using the Harbox dataset. Fig. 8 presents the results. *First*, AdaCLP shows the best balance between model accuracy and energy cost compared to all the baselines. *Second*, as illustrated in Fig. 8a, AdaCLP achieves a 4%–11% improvement in accuracy over the best FedCurv+CLP approaches. This improvement is primarily due to the CLP extension, which fine-tunes hyperparameters to mitigate overfitting during training with highly heterogeneous data streams. *Third*, as shown in Fig. 8b, AdaCLP reduces energy consumption by up to 12%, compared to FedAsyn and AFedCurv, the best AFedCurv, after 9 hours of training. This is because AdaCLP employs lower CPU frequencies during CLP, extending the duration of the process and minimizing energy usage.

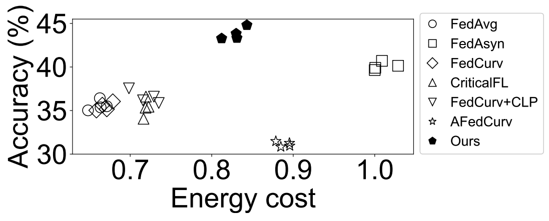
To further test robustness, we introduce *aperiodic data conditions*, simulating irregular class occurrences, such as rare or event-driven behaviors—by assigning each class C a distinct appearance probability across time. This breaks the



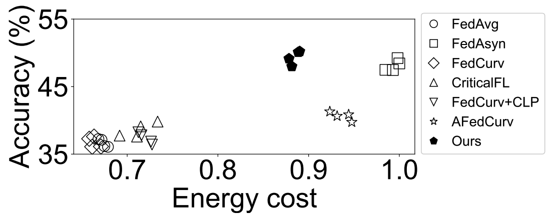
(a) 9 hours, Harbox, LSTM



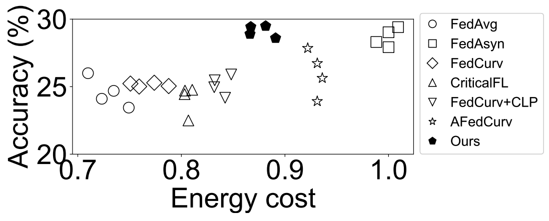
(b) 15 hours, Harbox, LSTM



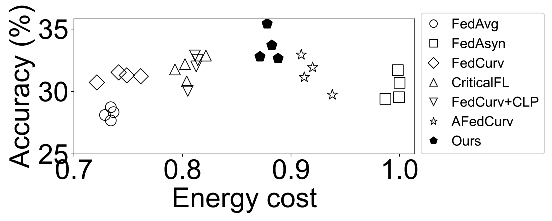
(c) 9 hours, HHAR, LSTM



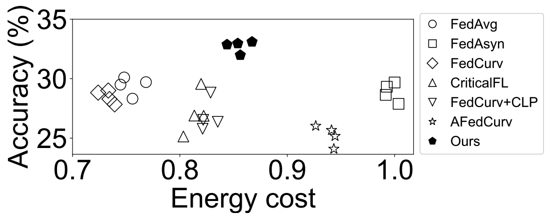
(d) 15 hours, HHAR, LSTM



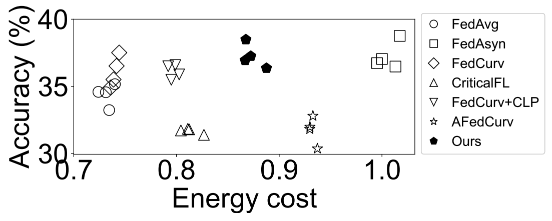
(e) 9 hours, CAUTURE24, LSTM



(f) 15 hours, CAPTURE24, LSTM



(g) 9 hours, CAPTURE24, Transformer



(h) 15 hours, CAPTURE24, Transformer

Fig. 9. Comparison of accuracy vs. energy cost between AdaCLP and other baselines in non-extreme setups.

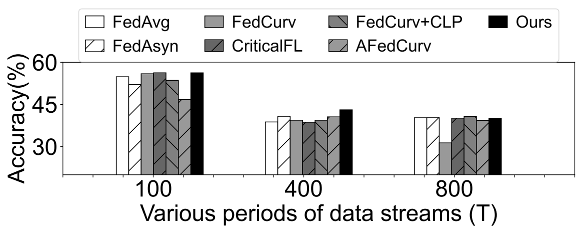
strong temporal periodicity assumed in many FL settings. Tab. 2 shows the results. AdaCLP remains more robust than all baselines under such irregular conditions, adapting learning dynamics via Fisher information–guided CLP control. While performance degrades as aperiodicity increases, AdaCLP maintains a significant advantage without relying on fixed timing assumptions or explicit class frequency heuristics.

Summary. AdaCLP outperforms six baselines in both accuracy and energy efficiency under extreme class imbalance and heterogeneity. Its ability to adapt to sporadic and non-periodic class arrivals further demonstrates its practicality for real-world mobile SFL applications in energy-constrained and temporally diverse environments.

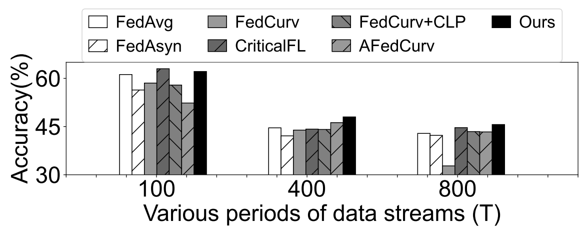
5.2.2 Performance Comparison in Non-Extreme Setups

We adopt a non-extreme setting as in Sec. 5.1, where the subset of classes C_i follows a Dirichlet distribution $\sim \mathrm{Dir}(\beta)$ with $\beta = 0.2$, and $T = 400$ is the period before switching to the next class. We measure the accuracy and energy consumption over 9-hour and 15-hour periods.

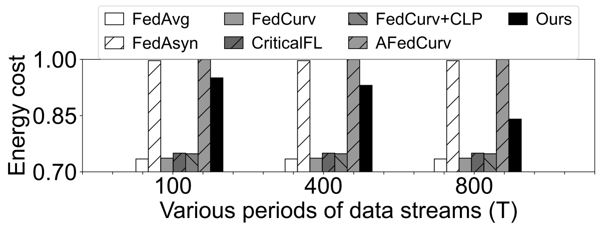
Fig. 9 shows the experimental results. *First*, AdaCLP exhibits the best overall performance in terms of accuracy and energy cost compared to all the baselines. While synchronous methods consume less energy, they result in lower accuracy within the same timeframe. In contrast, AdaCLP achieves the highest accuracy with the lowest energy cost by learning a sufficiently generalized base model with minimal energy within the CLP, while training quickly outside the CLP to reach high accuracy. *Second*, AdaCLP shows the highest accuracy across all datasets, outperforming all baseline methods. For example, as shown in Fig. 9g, AdaCLP achieves 1.57%–14.1% higher accuracy than the baselines. This is due to AdaCLP's ability to extend the Critical Learning Period (CLP), mitigating the global model generalization limitations caused by class imbalance, and allowing the model to learn more categories within the CLP. *Third*, AdaCLP exhibits the lowest energy consumption among all asynchronous baseline methods across three tasks. As shown in Fig. 9b, AdaCLP reduces energy consumption by 7% compared to FedAsyn and AFedCurv. However, AdaCLP's energy consumption is higher than that of synchronous methods in the same timeframe. This is due to



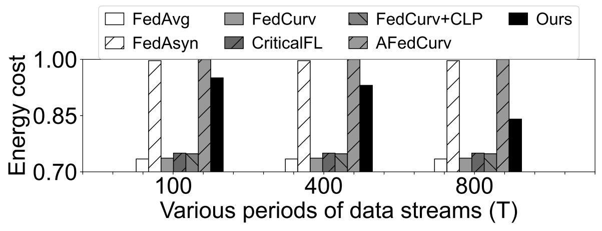
(a) Accuracy, 9 hours



(b) Accuracy, 15 hours



(c) Energy cost, 9 hours



(d) Energy cost, 15 hours

Fig. 10. Performance comparison with various periodic data streams.

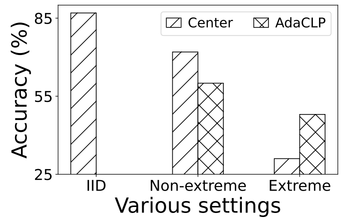


Fig. 11. Performance compared to centralized learning across different scenarios.

TABLE 2
Accuracy under extreme non-periodic settings.

Extreme Setting	Accuracy (%)
Uniform Class Probability	48.50%
One Class at 1% Probability	45.04%
Two Classes at 1% Probability	38.9%
Four Classes at 1% Probability	32.04%

the waiting time caused by device heterogeneity, leading to fewer total training rounds in synchronous methods compared to asynchronous ones.

Summary. AdaCLP strikes an optimal balance between accuracy and energy efficiency, leveraging the extended Critical Learning Period (CLP), adaptive hyperparameter tuning, and CLP-aware DVFS. In non-extreme scenarios, it achieves 14.11% higher accuracy and 12% lower energy cost compared to asynchronous baselines. This highlights AdaCLP's effectiveness in overcoming real-world *class imbalance* and *system heterogeneity*, making it a robust solution for federated learning in human activity recognition applications.

5.2.3 *Performance Comparison to Centralized Learning*

We compare AdaCLP and centralized learning under extreme and non-extreme conditions (Sec. 5.1). Centralized baselines include: *i) Centralized-Extreme* aggregates data from all 120 clients with a class switch every T=40 time units; *ii) Centralized-Non-extreme* uses the similar aggregation with moderately skewed data (β=0.3); *iii) Centralized-IID*: pooled IID data from all clients without any temporal or distributional skew. All centralized experiments use the same model architecture as AdaCLP but without communication or personalization constraints. As shown in Fig. 11, centralized achieves 87% in IID, drops to 72% in non-extreme, and 31% in extreme. AdaCLP achieves 60% (12% lower) in non-extreme, and 48% (17% higher) in extreme.

Summary. These results highlight AdaCLP's robustness to real-world dynamics such as class imbalance, temporal drift, and distributed heterogeneity, showing it can even outperform centralized training under highly skewed conditions.

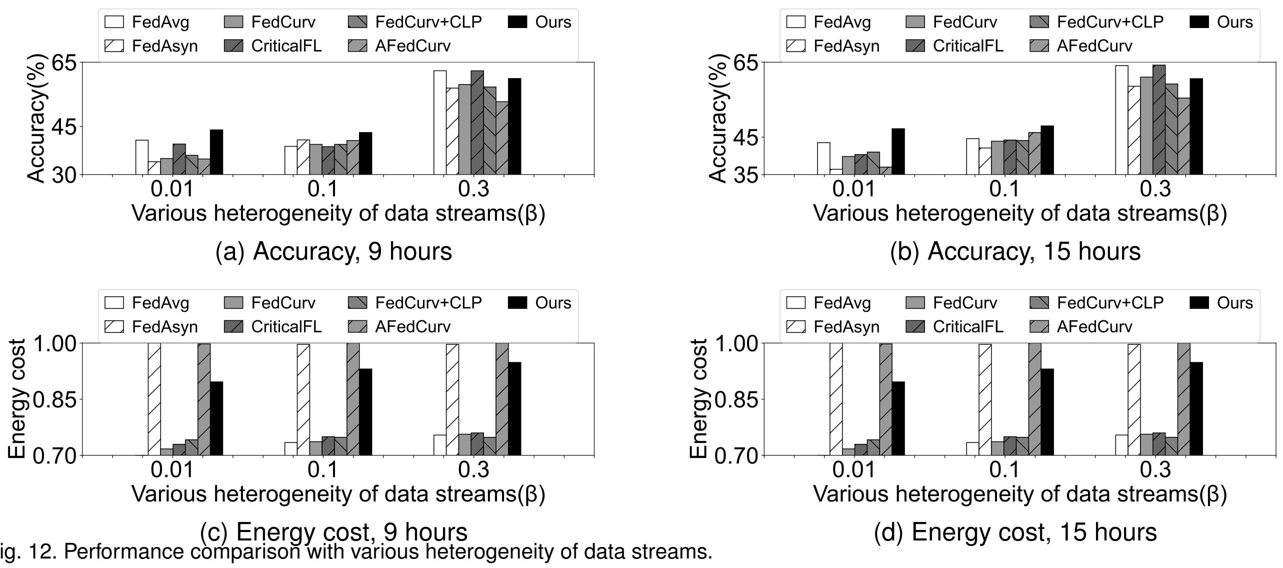
5.3 Performance in Various Scenarios

5.3.1 *Performance with Various Periodic Scheme of Data Streams*

We test AdaCLP and six baselines in four *periodic data stream* scenarios under realistic, *non-extreme* settings (Sec. 5.1), where class subsets $C_i \sim \text{Dir}(0.2)$ and period durations $T \in 100, 400, 800$ reflect different data dynamics. We measure accuracy and energy over 9-hour and 15-hour durations to capture temporal and environmental variations in mobile scenarios.

As shown in Fig. 10, *first*, AdaCLP achieves consistently higher accuracy, with the advantage increasing as T grows—outperforming baselines by up to 12.87% at $T = 800$ thanks to its adaptive CLP extension mitigating temporal imbalance. *Second*, AdaCLP reduces energy consumption more effectively as T increases, saving up to 7.84% at $T = 800$ via CLP-based DVFS optimizing idle training time.

Summary. AdaCLP effectively addresses long-period temporal class bias with the following advantages: *i)* Adaptive CLP extension enhances model plasticity, achieving the highest accuracy across varying periods. *ii)* CLP-based DVFS optimizes idle time for training, significantly reducing energy cost without sacrificing accuracy. These make AdaCLP ideal for energy-constrained mobile HAR FL systems with highly periodic data streams.



(a) Accuracy, 9 hours

(b) Accuracy, 15 hours

(c) Energy cost, 9 hours

(d) Energy cost, 15 hours

Fig. 12. Performance comparison with various heterogeneity of data streams.

TABLE 3
Settings of asynchronous (*i.e.*, fast and slow) devices

Index	Slow/Fast device quantity proportion	Slow/Fast device performance ratio
D1	1:1	1:5
D2	1:1	1:10
D3	1:1	1:15
D4	1:1	1:20

5.3.2 *Performance with Various Heterogeneity of Data Stream*

We evaluate AdaCLP and six baselines under *four heterogeneity levels* in both *non-extreme* and *extreme* settings (Sec. 5.1), with class subsets $C_i \sim \text{Dir}(\beta)$ ($\beta = 0.3, 0.1, 0.01$) and periodic duration $T = 400$. To reflect real-world variations, we measure accuracy and energy over 9-hour and 15-hour periods.

As shown in Fig. 12, *first*, AdaCLP achieves the highest accuracy except at $\beta = 0.3$, with its advantage increasing as β decreases—outperforming FedAvg by up to 3.76% at $\beta = 0.01$. *Second*, AdaCLP delivers more significant energy savings as heterogeneity increases, reaching up to 10.37% at $\beta = 0.01$, thanks to CLP-based DVFS continuously optimizing energy over long training periods.

Summary. AdaCLP addresses long-period temporal class bias with two main advantages: *i)* CLP extension maximizes global model plasticity, achieving superior accuracy across varying time periods; *ii)* CLP-based DVFS efficiently leverages idle time for training, reducing energy consumption without compromising accuracy.

5.3.3 *Performance over Heterogeneous Mobile Devices.*

We evaluate AdaCLP, FedAvg, and FedAsyn on *heterogeneous devices* under an *extreme* setting (Sec. 5.1), where C_i contains a single class and $T = 400$. Clients are evenly split into *fast* and *slow* devices, with training time ratios of 1:5, 1:10, 1:15, and 1:20 (Tab. 3), simulating real-world GPU/non-GPU heterogeneity. We measure the time to reach accuracy thresholds and the highest accuracy within fixed training time. We measure the time taken by AdaCLP and the baselines to achieve various accuracy thresholds within a 9-hour training window.

As shown in Fig. 13, *first*, AdaCLP achieves target accuracy fastest, reducing time by up to 69.5% under D1–D3 and outperforming FedAsyn by 23.87% even under extreme D4. *Second*, AdaCLP consistently reaches higher accuracy within 15 hours, achieving 45% accuracy in 263–427 minutes under D1–D3, which neither baseline achieves.

Summary. AdaCLP effectively addresses the challenges posed by device heterogeneity. By leveraging a staleness-decayed CLP measurement, it accurately evaluates and extends the CLP, mitigating the impact of temporal class bias on accuracy.

5.3.4 *Energy Cost on CPU-only and CPU+GPU Devices*

We evaluate AdaCLP against FedAvg and FedAsyn on CPU-only (Raspberry Pi 4) and CPU+GPU (Jetson Xavier NX) devices under an *extreme* setting (Sec. 5.1), with a single-class C_i and period $T = 400$. Energy cost to reach target accuracy within 15 hours is measured.

As shown in Fig. 14, *first*, on CPU-only devices, AdaCLP achieves the target accuracy fastest and with the lowest energy, cutting energy by 44.82% vs. FedAvg and 57.09% vs. FedAsyn (Fig. 14b). *Second*, similar results hold for CPU+GPU, with energy savings of 44.71% and 80.12%, respectively (Fig. 14d), thanks to CLP-based DVFS exploiting idle time for significant savings.

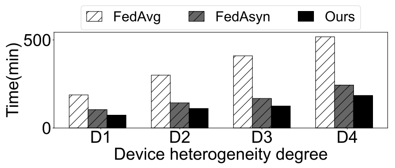
5.4 Micro-benchmark and Ablation Study

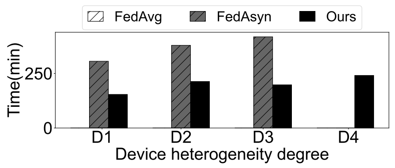
5.4.1 *Impact of Batch Size*

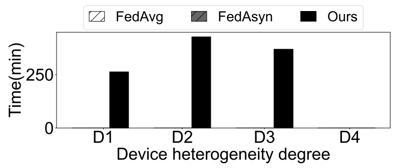
We test AdaCLP with batch sizes of 10, 30, and 50 under an *extreme* setting (Sec. 5.1), with a single-class C_i and period $T = 400$, to assess the impact of initial batch size. As shown in Fig. 15a, accuracy improves from 45.27% to 47.98% as batch size increases, since larger batches enable learning more data within the same time. We recommend a batch size of about 50 for optimal performance.

5.4.2 *Impact of Learning Rate*

We evaluate AdaCLP with learning rates of 0.01, 0.05, and 0.001 under an *extreme* setting (Sec. 5.1), with single-class C_i and period $T = 400$. As shown in Fig. 15b, accuracy

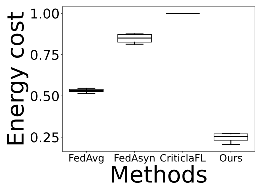


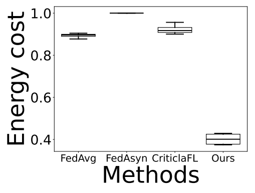


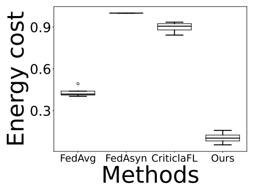


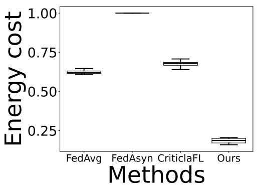
(a) Convergence time for 30% accuracy (b) Convergence time for 40% accuracy (c) Convergence time for 45% accuracy

Fig. 13. Convergence time comparison with various heterogeneity of mobile devices.



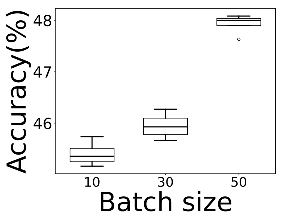


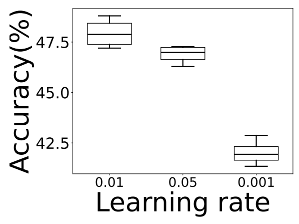


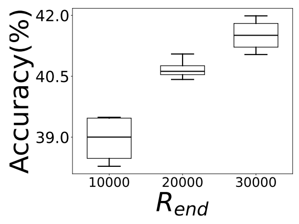


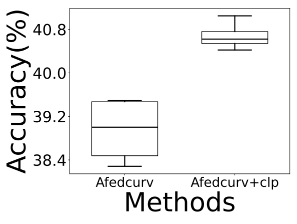
(a) Energy cost for 35% accuracy with CPU (b) Energy cost for 40% accuracy with CPU (c) Energy cost for 35% accuracy with CPU+GPU (d) Energy cost for 40% accuracy with CPU+GPU

Fig. 14. Energy cost to achieve different accuracy threshold.









(a) Impact of batch size (b) Impact of learning rate (c) Impact of R_{end} (d) Impact of CLP regulator

Fig. 15. Micro-benchmark and ablation study.

improves from 41.74% to 48.31% as the learning rate increases, since larger rates allow faster adaptation via CLP. We recommend a learning rate of 0.01 for best performance.

5.4.3 Maximum round R_{end} of CLP Extension

We evaluate AdaCLP with maximum rounds R_{end} of 10000, 20000, and 30000 under an *extreme* setting (Sec. 5.1), using single-class C_i and period $T = 400$, to assess the impact of CLP extension. As shown in Fig. 15, increasing R_{end} raises accuracy from 39.49% to 41.73%, highlighting the benefit of longer CLP extension. We recommend setting R_{end} to 30000 for optimal performance.

5.4.4 Effectiveness of federated CLP regulator

We evaluated the Federated CLP Regulator's orthogonal feasibility by integrating it with AFedCurv under an *extreme* setting (Sec. 5.1), using single-class C_i and period $T = 400$. As shown in Fig. 15d, applying the CLP Regulator improved AFedCurv's accuracy by 2.29%, confirming its positive, non-intrusive effect when combined with other asynchronous methods.

5.4.5 Computation Overhead

We measure the per-client training time with/without FIM on Harbox, using Transformer and LSTM across 120 clients. Fig. 17 shows FIM adds <40% overhead, which drops from 40% (LSTM) to 32% (Transformer) as model size increases, since training time grows faster than FIM cost.

TABLE 4
Trade-off of Class Diversity vs. Local Updates in CLP-aware DVFS

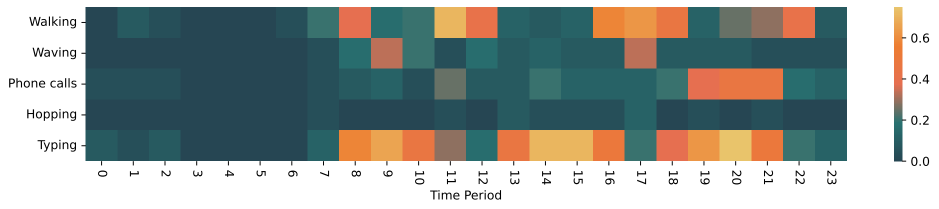
Methods	Accuracy(%)
FedAsync	38.78
AdaCLP (Fixed at lowest frequency)	42.43
AdaCLP (CLP-aware DVFS)	43.93

5.4.6 Class Diversity vs. Local Updates in CLP-aware DVFS

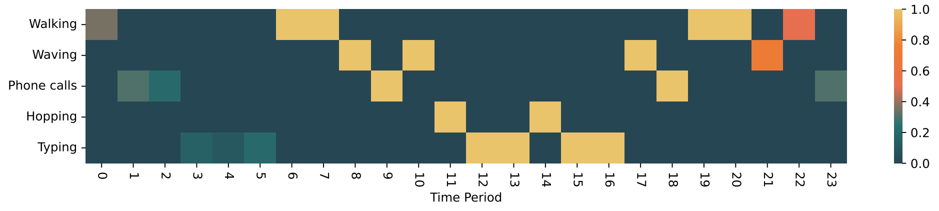
AdaCLP uses DVFS to lower CPU/GPU frequency during CLP, enhancing early class diversity but reducing rounds. We compare three cases: (i) FedAsync, (ii) AdaCLP with fixed low frequency, (iii) AdaCLP with dynamic frequency. As Table 6 shows, FedAsync performs worst, and dynamic AdaCLP improves over fixed by 1.5% due to better late-stage convergence. This demonstrates that adaptive scheduling effectively balances early diversity and late training depth, improving final performance.

5.4.7 Sensitivity analysis of λ

We evaluate the staleness decay parameter λ in our CLP-based scheme (Eq. 2) under an extreme scenario with $\lambda \in \{1, 0.5, 0.1, 0.05, 0.01, 0.001\}$. As shown in Fig. 18, λ in $[0.01, 0.1]$ achieves high accuracy and robustness, while extreme values (near 1 or 0.001) cause instability and performance drop. We set $\lambda = 0.01$ by default.



(a) First day: mobile data streams with *non-extreme* data class shifts



(b) Second day: mobile data streams with *extreme* data class shifts

Fig. 16. We employed 20 users, each using different mobile devices to simulate an asynchronous Federated Learning (FL) setup, with users subjected to different non-extreme and extremely personalized data category shifts over two days.

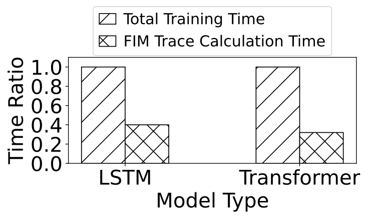


Fig. 17. Time Ratio of Calculating vs. Not Calculating FIM Trace.

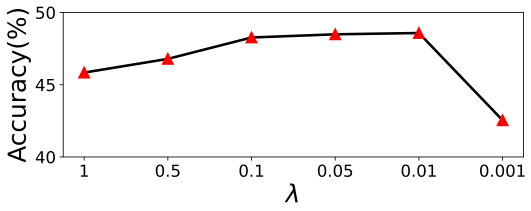


Fig. 18. Comparison of accuracy at different λ

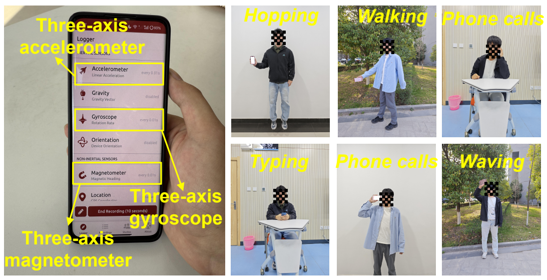


Fig. 19. Illustration of user data collection and test.

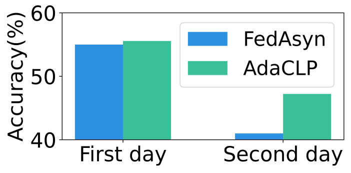


Fig. 20. Average accuracy across 20 mobile devices.

5.5 Case Study

We conducted a two-day study, as shown on the left side of Fig. 19, using a free smartphone application (Sensor Logger app) to collect acceleration data from 20 participants. These participants included 5 engineering undergraduates, 5 sports students, 5 engineering lecturers, and 5 restaurant staff. Following the data collection approach of the HarBox dataset [39], participants carried their smartphones during daily activities to capture five types of behaviors: Walking, Hopping, Phone Calls, Waving, and Typing. As on the right side of Fig. 19, we present examples of several actions performed by three participants. Each device stored nine-axis inertial measurement unit (IMU) data over two days. Additionally, 5 volunteers performed standardized actions from the training dataset 10 times to create a benchmark test set.

To evaluate AdaCLP 's performance under varying settings, participants engaged in free activities on the first day to simulate *non-extreme* temporal category biases. On the second day, they followed guided tasks at specific times to simulate *extreme* temporal category biases. Fig. 16 illustrates the temporal shifts over the two days. Fig. 20 shows that, compared to FedAsyn, AdaCLP enabled mobile client models to better adapt to temporal category biases. Meanwhile, the global model maintained stable accuracy (*i.e.*, 6.22% higher than FedAsyn) despite extreme data biases, demonstrating AdaCLP 's effectiveness in handling streaming data scenarios with temporal class imbalance bias.

6 Related Work

FL in Mobile Applications. Federated learning (FL) has been widely adopted in mobile applications, such as activity recognition [24], [25], [39], personalized recommendations [22], [26], [27], [29], and intelligent transportation systems [28], [33], [36], among others. Deploying FL on mobile devices (*e.g.*, smartphones, wearables) enhances data privacy and minimizes communication overhead during the training of DL models. Generic FL frameworks typically aim to train a *single* global model for all clients, with FedAvg [2] being the standard method. To address *data heterogeneity* across diverse clients, various enhancements have been proposed, *e.g.*, regularization [15], [16], [55] and hyperparameter control [35], [41] to mitigate gradient divergence caused by non-IID data at the client side. And server-side approaches include weight matching [68] and knowledge distillation [6], [7], [61] to improve model aggregation. Despite these advancements, most FL solutions assume *pre-collected data* on mobile clients, overlooking real-world scenarios where data arrives in *streams* [21], [54]. And they often *exhibit diverse periodic patterns and heterogeneity*, further complicated by *asynchronous* model updates across devices with varying computational capacities [29], [64].

Memory-efficient Streaming Federated Learning with Live Data Streams. Streaming Federated Learning (SFL) reduces memory requirements and training latency via *immediate* local training and *asynchronous* aggregation. However, such immediacy often causes overfitting to temporal data distributions. Existing works mitigate this through *historical data retrieval* [3], [38], [52], [62] or *weight regularization* [5], [18], [54]. The former alleviates temporal overfitting by combining past and current data but increases storage cost, while the latter controls local update divergence to improve generalization without storage overhead [54]. AdaCLP achieves similar benefits without explicit regularization, offering a lightweight solution for constrained devices. Recent studies also explore streaming and mobile FL from complementary angles. Hu *et al.* [71] employ a Lyapunov drift-plus-penalty framework for non-periodic data streams, jointly optimizing scheduling and energy efficiency. Chen *et al.* [72] leverage Hierarchical Federated Learning (HFL) to show that mobility enhances convergence and accuracy by diversifying data and accelerating fusion. In contrast, AdaCLP focuses on periodic user data with strong temporal skew (e.g., time-of-day behavior patterns), formally introduces temporal class imbalance, and employs a plasticity-guided, energy-efficient learning mechanism to mitigate early overfitting and improve generalization under heterogeneous mobile settings.

Critical Learning Period-aware Learning. The Critical Learning Period (CLP) refers to specific phases in the training of DNNs that are highly susceptible to overfitting, particularly in the presence of non-IID data [12], [42]. During the early training stages, DNNs are especially vulnerable to overfitting, which can result in irrecoverable accuracy degradation [1], [32]. Originally introduced for centralized training, the concept of CLP has recently been extended to federated learning (FL). For instance, Yan *et al.* [1] utilize the Fisher Information Matrix traces of the global model to quantify CLP in FL. Additionally, incorporating more clients during this critical phase has been shown to mitigate non-IID effects and reduce overfitting [32]. However, existing methods focus on *quantifying* and *exploiting CLP* within *synchronous* FL settings, leaving asynchronous scenarios, such as Streaming Federated Learning (SFL), unaddressed. In contrast, AdaCLP intends to adapt and leverage the CLP specifically for *asynchronous* real-world mobile environments, offering a novel perspective tailored to the challenges of SFL.

DVFS on Mobile Devices. Dynamic Voltage and Frequency Scaling (DVFS) is a widely adopted technique that dynamically adjusts processor voltage and frequency on mobile devices to reduce energy consumption and manage thermal effects [10], [34], [50], [60], [63]. For DNN applications, existing DVFS strategies primarily target at achieving lower latency while avoiding device overheating by determining the thermal throttling boundaries of mobile devices through deep reinforcement learning combined with transfer learning [34], [40], and minimizing energy consumption with minimal performance degradation via adaptive sensor power gating units [31], [45], [59]. In the context of *streaming FL* tasks, which operate *intermittently*, our focus in AdaCLP is on optimizing energy efficiency rather than mitigating thermal issues, aligning with the sporadic nature of these workloads.

7 Conclusion

In this paper, we address the temporal class imbalance problem in streaming federated learning (SFL) via AdaCLP, a novel plasticity-aware training scheduler. By extending the critical learning period (CLP) of deep neural networks, AdaCLP enhances model adaptability and accuracy without relying on additional data storage or modifications to standard training processes. The proposed federated CLP regulator ensures accurate plasticity measurements even in asynchronous settings through a staleness-decayed averaging mechanism. Additionally, the integration of dynamic voltage frequency scaling (DVFS) mitigates the energy overhead of prolonged CLPs, making AdaCLP a practical solution for resource-constrained mobile devices. Empirical results on 20 mobile devices demonstrate that AdaCLP achieves notable improvements in accuracy and energy efficiency. It establishes AdaCLP as a robust framework for SFL, paving the way for collaborative and continual learning on mobile devices. In the future, we plan to integrate AdaCLP with more advanced asynchronous FL algorithms and extend it for training personalized models.

Acknowledgment

This work was partially supported by the National Science Fund for Distinguished Young Scholars (62025205) and the National Natural Science Foundation of China (No. 62522215,62532009,62472354)

References

[1] 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.
[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] Marfoq O, Neglia G, Kameni L, et al. Federated learning for data streams[C]//International Conference on Artificial Intelligence and Statistics. PMLR, 2023: 8889-8924.
[4] Chen Y, Ning Y, Slawski M, et al. Asynchronous online federated learning for edge devices with non-iid data[C]//2020 IEEE International Conference on Big Data (Big Data). IEEE, 2020: 15-24.
[5] Dong J, Wang L, Fang Z, et al. Federated class-incremental learning[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022: 10164-10173.
[6] 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.

[7] 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.

[8] Huang P, Kumar P, Giannopoulou G, et al. Energy efficient DVFS scheduling for mixed-criticality systems[C]//Proceedings of the 14th International Conference on Embedded Software. 2014: 1-10.

[9] Mei X, Yung L S, Zhao K, et al. A measurement study of GPU DVFS on energy conservation[C]//Proceedings of the Workshop on Power-Aware Computing and Systems. 2013: 1-5.

[10] Kim W, Gupta M S, Wei G Y, et al. System level analysis of fast, per-core DVFS using on-chip switching regulators[C]//2008 IEEE 14th International Symposium on High Performance Computer Architecture. IEEE, 2008: 123-134.

[11] Jastrzębski S, Kenton Z, Ballas N, et al. On the relation between the sharpest directions of DNN loss and the SGD step length[J]. arXiv preprint arXiv:1807.05031, 2018.

[12] Achille A, Rovere M, Soatto S. Critical learning periods in deep networks[C]//International Conference on Learning Representations. 2018.

[13] Shang X, Lu Y, Cheung Y, et al. Fedic: Federated learning on non-iid and long-tailed data via calibrated distillation[C]//2022 IEEE International Conference on Multimedia and Expo (ICME). IEEE, 2022: 1-6.

[14] Eichner H, Koren T, McMahan B, et al. Semi-cyclic stochastic gradient descent[C]//International Conference on Machine Learning. PMLR, 2019: 1764-1773.

[15] 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.

[16] 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.

[17] Yoon J, Jeong W, Lee G, et al. Federated continual learning with weighted inter-client transfer[C]//International Conference on Machine Learning. PMLR, 2021: 12073-12086.

[18] Yao X, Sun L. Continual local training for better initialization of federated models[C]//2020 IEEE International Conference on Image Processing (ICIP). IEEE, 2020: 1736-1740.

[19] Ma Y, Xie Z, Wang J, et al. Continual Federated Learning Based on Knowledge Distillation[C]//IJCAI. 2022: 2182-2188.

[20] Zhu C, Xu Z, Chen M, et al. Diurnal or nocturnal? federated learning of multi-branch networks from periodically shifting distributions[C]//International Conference on Learning Representations. 2021.

[21] Wei Y, Xiong J, Liu H, et al. AdaStreamLite: Environment-adaptive Streaming Speech Recognition on Mobile Devices[J]. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 2024, 7(4): 1-29.

[22] Guo Y, Liu F, Cai Z, et al. PREFER: Point-of-interest REcommendation with efficiency and privacy-preservation via Federated Edge leaRning[J]. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 2021, 5(1): 1-25.

[23] Zhang M, Sawchuk A A. USC-HAD: A daily activity dataset for ubiquitous activity recognition using wearable sensors[C]//Proceedings of the 2012 ACM conference on ubiquitous computing. 2012: 1036-1043.

[24] 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.

[25] Gong K, Gao Y, Dong W. Privacy-preserving and cross-domain human sensing by federated domain adaptation with semantic knowledge correction[J]. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 2024, 8(1): 1-26.

[26] 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.

[27] Liu S, Guo B, Ma K, et al. AdaSpring: Context-adaptive and runtime-evolutionary deep model compression for mobile applications[J]. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 2021, 5(1): 1-22.

[28] Tabatabaie M, He S. Driver maneuver interaction identification with anomaly-aware federated learning on heterogeneous feature representations[J]. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 2024, 7(4): 1-28.

[29] 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.

[30] 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.

[31] Hou X, Liu J, Tang X, et al. Architecting efficient multi-modal aiot systems[C]//Proceedings of the 50th Annual International Symposium on Computer Architecture. 2023: 1-13.

[32] 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.

[33] Yang L, Chen W, He X, et al. FedGTP: Exploiting inter-client spatial dependency in federated graph-based traffic prediction[C]//Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. 2024: 6105-6116.

[34] Lin C, Wang K, Li Z, et al. A workload-aware DVFS robust to concurrent tasks for mobile devices[C]//Proceedings of the 29th Annual International Conference on Mobile Computing and Networking. 2023: 1-16.

[35] 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.

[36] Zheng T, Li A, Chen Z, et al. Autofed: Heterogeneity-aware federated multimodal learning for robust autonomous driving[C]//Proceedings of the 29th annual international conference on mobile computing and networking. 2023: 1-15.

[37] Yue S, Ren J, Xin J, et al. Inexact-ADMM based federated meta-learning for fast and continual edge learning[C]//Proceedings of the Twenty-second International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing. 2021: 91-100.

[38] Zhang R, Zou Y, Xie Z, et al. Federating from History in Streaming Federated Learning[C]//Proceedings of the Twenty-fifth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing. 2024: 151-160.

[39] 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.

[40] Kim S, Bin K, Ha S, et al. zTT: Learning-based DVFS with zero thermal throttling for mobile devices[J]. GetMobile: Mobile Computing and Communications, 2022, 25(4): 30-34.

[41] 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.

[42] Tirumala K, Markosyan A, Zettlemoyer L, et al. Memorization without overfitting: Analyzing the training dynamics of large language models[J]. Advances in Neural Information Processing Systems, 2022, 35: 38274-38290.

[43] Mirzadeh S I, Farajtabar M, Pascanu R, et al. Understanding the role of training regimes in continual learning[J]. Advances in Neural Information Processing Systems, 2020, 33: 7308-7320.

[44] You J, Chung J W, Chowdhury M. Zeus: Understanding and optimizing GPU energy consumption of DNN training[C]//20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23). 2023: 119-139.

[45] Hou X, Tang P, Li C, et al. Smg: A system-level modality gating facility for fast and energy-efficient multimodal computing[C]//2023 IEEE Real-Time Systems Symposium (RTSS). IEEE, 2023: 291-303.

[46] Chou Y H, Hong S, Sun C, et al. Grp-fed: Addressing client imbalance in federated learning via global-regularized personalization[C]//Proceedings of the 2022 SIAM International Conference on Data Mining (SDM). Society for Industrial and Applied Mathematics, 2022: 451-458.

[47] Stisen A, Blunck H, Bhattacharya S, et al. Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition[C]//Proceedings of the 13th ACM conference on embedded networked sensor systems. 2015: 127-140.

[48] Fang C, Liu S, Zhou Z, et al. AdaShadow: Responsive test-time model adaptation in non-stationary mobile environ-

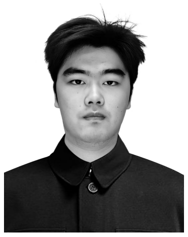
ments[C]//Proceedings of the 22nd ACM Conference on Embedded Networked Sensor Systems. 2024: 295-308.
[49] Wu F, Liu S, Zhu K, et al. AdaFlow: Opportunistic Inference on Asynchronous Mobile Data with Generalized Affinity Control[C]//Proceedings of the 22nd ACM Conference on Embedded Networked Sensor Systems. 2024: 606-618.
[50] 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, 26(1): 389-427.
[51] Niknejad N, Ismail W B, Mardani A, et al. A comprehensive overview of smart wearables: The state of the art literature, recent advances, and future challenges[J]. Engineering Applications of Artificial Intelligence, 2020, 90: 103529.
[52] Wang H, Bian J, Xu J. On the local cache update rules in streaming federated learning[J]. IEEE Internet of Things Journal, 2023, 11(6): 10808-10816.
[53] Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: a simple way to prevent neural networks from overfitting[J]. The journal of machine learning research, 2014, 15(1): 1929-1958.
[54] Jin Y, Jiao L, Qian Z, et al. Budget-aware online control of edge federated learning on streaming data with stochastic inputs[J]. IEEE Journal on Selected Areas in Communications, 2021, 39(12): 3704-3722.
[55] 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.
[56] Kirkpatrick J, Pascanu R, Rabinowitz N, et al. Overcoming catastrophic forgetting in neural networks[J]. Proceedings of the national academy of sciences, 2017, 114(13): 3521-3526.
[57] Rizvandi N B, Taheri J, Zomaya A Y. Some observations on optimal frequency selection in DVFS-based energy consumption minimization[J]. Journal of Parallel and Distributed Computing, 2011, 71(8): 1154-1164.
[58] Chan S, Hang Y, Tong C, et al. CAPTURE-24: A large dataset of wrist-worn activity tracker data collected in the wild for human activity recognition[J]. Scientific Data, 2024, 11(1): 1135.
[59] Hou X, Xu C, Li C, et al. Improving Efficiency in Multi-modal Autonomous Embedded Systems through Adaptive Gating[J]. IEEE Transactions on Computers, 2024.
[60] Liu S, Du J, Nan K, et al. AdaDeep: A usage-driven, automated deep model compression framework for enabling ubiquitous intelligent mobiles[J]. IEEE Transactions on Mobile Computing, 2020, 20(12): 3282-3297.
[61] Li X, Liu S, Zhou Z, et al. ClassTer: Mobile Shift-Robust Personalized Federated Learning via Class-Wise Clustering[J]. IEEE Transactions on Mobile Computing, 2024.
[62] Wang N, Li X, Guan Z, et al. Fedstream: A federated learning framework on heterogeneous streaming data for next-generation traffic analysis[J]. IEEE Transactions on Network Science and Engineering, 2023, 11(3): 2485-2496.
[63] Nabavinejad S M, Reda S, Ebrahimi M. Coordinated batching and DVFS for DNN inference on GPU accelerators[J]. IEEE transactions on parallel and distributed systems, 2022, 33(10): 2496-2508.
[64] Xie C, Koyejo S, Gupta I. Asynchronous federated optimization[J]. arXiv preprint arXiv:1903.03934, 2019.
[65] Shang X, Lu Y, Huang G, et al. Federated learning on heterogeneous and long-tailed data via classifier re-training with federated features[J]. arXiv preprint arXiv:2204.13399, 2022.
[66] Xu C, Hong Z, Huang M, et al. Acceleration of federated learning with alleviated forgetting in local training[J]. arXiv preprint arXiv:2203.02645, 2022.
[67] Shoham N, Avidor T, Keren A, et al. Overcoming forgetting in federated learning on non-iid data[J]. arXiv preprint arXiv:1910.07796, 2019.
[68] Wang H, Yurochkin M, Sun Y, et al. Federated learning with matched averaging[J]. arXiv preprint arXiv:2002.06440, 2020.
[69] Mishra S M. Wearable android: android wear and google fit app development[M]. John Wiley & Sons, 2015.
[70] North F, Chaudhry R. Apple healthkit and health app: patient uptake and barriers in primary care[J]. Telemedicine and e-Health, 2016, 22(7): 608-613.
[71] C. -H. Hu, Z. Chen and E. G. Larsson, "Energy-Efficient Federated Edge Learning With Streaming Data: A Lyapunov Optimization Approach," in IEEE Transactions on Communications, vol. 73, no. 2, pp. 1142-1156, Feb. 2025, doi: 10.1109/TCOMM.2024.3443731.
[72] T. Chen, J. Yan, Y. Sun, S. Zhou, D. Gündüz and Z. Niu, "Mobility Accelerates Learning: Convergence Analysis on Hierarchical Federated Learning in Vehicular Networks," in IEEE Transactions on Vehicular Technology, vol. 74, no. 1, pp. 1657-1673, Jan. 2025, doi: 10.1109/TVT.2024.3466299.

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.

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).

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).

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.



Weiye Wu is a Master's student at NorthwesternPolytechnical University, China. 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 the Vice President of Harbin Engineering University and a Professor at the School of Computer Science, Northwestern Polytechnical University. 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.