A Continuous-Time On-Device Federated Learning Network

Yimin Dai *Student Member, IEEE*, Rui Tan *Senior Member, IEEE*

***Abstract*—Time series is an important form of data generated by Internet-of-Things (IoT) devices. Closed-form continuous-time (CFC) neural networks offer superior expressivity for modeling time series data compared with recurrent neural networks. Additionally, their lower training and inference overhead make them well-suited for deployment on microcontroller-based IoT devices. This paper introduces FedCFC, an on-device federated learning network that operates based on the CFC models distributed across IoT devices. FedCFC incorporates a novel and communication-efficient aggregation strategy designed to mitigate the effects of class distribution imbalances across the participating IoT devices' training data. The strategy is designed based on a new property of CFC identified in this paper, i.e., the insensitivity of a sub-model of CFC with respect to training data's class distribution shift. Extensive evaluation with multiple time series datasets demonstrates that FedCFC attains comparable or superior accuracy while achieving a 7.6× to 11× reduction in communication overhead compared with recent federated learning approaches designed to address the class distribution skew problem. Furthermore, deployments of FedCFC on four IoT platforms highlight its suitability for resource-constrained devices with as little as 256 kB of memory or even less.**

***Index Terms*—Continuous-time neural network, federated learning, IoT, on-device machine learning.**

I. Introduction

ADVANCING a data collection-oriented Internet-of-Things (IoT) network to a data processing network by utilizing IoT devices' computing capabilities is a desirable move, due primarily to the high energy overhead of transmitting raw data and the application constraints such as data privacy. Among the various processing tasks, modeling and inference from time series data using deep neural networks have received significant attention [1], [2]. As time series data is temporally structured, the processing models should capture temporal dependencies effectively. However, conventional feedforward neural networks (FNNs) primarily focus on static input-output mappings and do not explicitly model the temporal dimension. For instance, a neuron in a multilayer perceptron (MLP) represents the relationship between inputs and outputs in a steady-state manner, failing to account for the transient dynamics that occur between two steady states. To address this limitation, recurrent neural networks (RNNs) incorporate feedback mechanisms to introduce *neural network-level dynamics*, thereby improving performance on time series modeling tasks. Despite their advantages, RNNs operate under the assumption of discrete time steps with a fixed interval, which presents challenges in real-world applications where time series data may exhibit irregular sampling rates. Consequently, heuristic preprocessing techniques such as up/down-sampling and interpolation are often required to align the data with the model's assumptions. These preprocessing steps not only introduce computational overhead but may also lead to information loss or distortions in the data representation.

This paper considers continuous-time neural networks (CT-NNs) [3] that have *neuron-level dynamics* for time series data modeling and processing at the IoT devices. Moreover, this paper aims to develop the first continuous-time federated learning network, in which each IoT device learns from its local data with a CT-NN and exchanges essential CT-NN update information with a *parameter server*. The behavior of a neuron in a CT-NN is governed by an ordinary differential equation (ODE), where time is treated as an independent and explicit variable. Since the ODE incorporates trainable parameters, the neuron can dynamically adapt its internal states throughout the learning process, enabling a more flexible representation of temporal dependencies. Furthermore, CT-NNs support *neural network-level feedback*, allowing the model to capture both local neuron dynamics and global temporal structures in the data. This synergistic integration of the dynamics at the two levels enhances CT-NNs' expressivity, making them more effective than RNNs in learning complex time series patterns [4]. Moreover, due to CT-NNs' continuous-time formulation, they can operate on any timestamped time series data without imposing the requirement of regular sampling.

Despite these advantages, the deployment of CT-NNs is often constrained by computational bottlenecks. In particular, most ODEs lack closed-form solutions, necessitating the use of numerical ODE solvers to propagate the neural network state over time. These solvers introduce a significant computational overhead, as the cost of forward inference is proportional to the square of the number of prediction steps. In typical implementations, this results in CT-NNs requiring 50× more computation per training round than RNNs [5]. Encouragingly, a recent study [6] obtains a closed-form approximate solution to the ODE used by a class of CT-NNs called *liquid time constant* (LTC) neural networks [4]. It also designs a *closed-form continuous-time* (CFC) neural network based on three FNN modules to approximate the closed-form solution while preserving explicit consideration of continuous time, in that the CFC network jointly processes the time series data and the associated timestamps. Forwarding a CFC has a compute overhead proportional to the prediction steps, which is typically two orders lower than those of ODE solvers.

CFC is promising for addressing two system challenges faced by the implementations of machine learning algorithms/models at resource-constrained IoT devices. First, its native ability to process any timestamped time series data addresses various system issues, including data intermittency due to hardware/software faults, sampling interval jitters due to uncertain operating system delays, and varied sampling rates across different IoT devices. Second, CFC's superior model

Yimin Dai and Rui Tan are with College of Computing and Data Science, Nanyang Technological University, Singapore.

expressivity implies a smaller memory footprint for a certain accuracy level, thereby reducing the resource usage of model training and making on-device learning readily implementable. By combining these two strengths, CFC can process local time series data in real-time directly on IoT devices, which is critical for applications that require immediate responses, such as early detection of anomalies in industrial machinery operations (e.g., vibration-based fault detection in manufacturing).

In this paper, we design a continuous-time on-device federated learning (FL) network called FedCFC, where the participating IoT devices learn and run their CFC models. FL enables neural network training on distributed client data while preserving privacy by keeping raw data local. However, conventional FL methods impose significant computational and communication costs, posing challenges for deployment on resource-limited IoT devices. A preliminary study [7] shows that FL can cause two orders of magnitude more carbon emissions than centralized learning. CFC offers the following direct advantages in reducing FL's overheads. First, owing to the lower resource requirements of CFC training, we may push the frontier of FL applications from the high-end edge computing devices (e.g., smartphones) to low-end IoT devices based on microcontroller units (MCUs). Second, CFC's smaller model size reduces FL's communication overhead. Note that FL can be applied under the continual learning setting to update the IoT devices' models with new data distributed on these devices. If the devices are battery-based (e.g., wearable sensors), the model update computation can be scheduled during the charging cycles.

FedCFC requires a logic star network, where the parameter server is at the center of the network and the participating IoT devices maintain connections with the parameter server during the learning phase. It can operate on top of any communication infrastructure and network topology. This paper primarily considers a challenge from the data domain, i.e., the clients' data classes are not independent and identically distributed (non-IID). This can be observed in many IoT sensing applications. For instance, in a human activity recognition application, the distribution of the inertial data samples among the activity classes generally varies with the user. Similarly, in a voice-based keyword spotting system, different users may contribute recordings of only a subset of words. Such class distribution skews can lead to substantial oscillations in the loss trajectory and slow down the convergence when the vanilla FL aggregation strategy FedAvg [8] is adopted. FedCFC develops a new aggregation design for this non-IID setting. It is based on a property of CFC identified in this paper, i.e., one of the three CFC's FNN modules learns task-wide common pattern and is insensitive to the training data's class distribution shift. Therefore, FedCFC limits the scope of FL to this FNN module only and leaves the other two FNN modules unfederated to capture personal patterns. This design effectively balances the needs of capturing commonality and retaining personality under the non-IID setting. Moreover, FedCFC proposes a novel geometric aggregation strategy called *fBind*, in which only the loss gradient with respect to the last layer of the selected FNN module is exchanged. This further reduces communications.

The main contributions of this paper are as follows:

- We propose a revised CFC model with improved numerical stability. Our benchmark study and analysis identify the insensitivity of a key module of CFC with respect to the class distribution of training data. Based on the insensitivity, we design FedCFC to federate the key module only, with a new geometric aggregation algorithm fBind to efficiently address the non-IID data problem.
- Evaluation shows that the FedCFC with fBind outperforms the FedCFC variants using aggregation strategies of FedAvg [8], FedProx [9], and SCAFFOLD [10], where the latter two were designed for the non-IID setting. On three time series datasets, FedCFC achieves higher or similar accuracy with 7.6× to 14× reduction in communication overhead, compared with LotteryFL [11] and BalanceFL [12].
- Our implementations of FedCFC with communication module on four MCU-based IoT platforms show its less than 256 kB memory footprint and thus its portability to low-end devices. Code is released at https://github.com/hhhddyy/FedCFC

Paper organization: §II presents background. §III presents benchmarks and analysis for our improved CFC model. §IV, §V, and §VI present design, evaluation, and on-device experiments of FedCFC. §VII concludes this paper.

II. Background

This section presents the related work in §II-A and then the preliminaries about LTC [4] and CFC [6] in §II-B.

A. Related Work

1) On-device training: While on-device training capability is essential to federated learning and continual learning, it is challenging for resource-constrained IoT devices. MDLdroid [13] manages model structures and learning process to achieve on-device training on smartphones with gigabytes dynamic random access memory (DRAM). However, typical MCUs have limited static random access memory (SRAM), in the range of 10^0 kB to 10^3 kB. Therefore, on-device training on MCUs is challenging. A survey [14] reviews the research works on machine learning topics for MCU-class hardware, including feature engineering, model design, and on-device training. It points out that the existing on-device training studies either consider high-end edge devices such as Raspberry Pi or simply limit the training to a few layers to meet the resource constraints. Some recent efforts, such as [15], have demonstrated the feasibility of on-device centralized training within 256 kB SRAM by leveraging memory-efficient deep learning techniques. This work will show that, with CFC, on-device FL with 256 kB SRAM can be achieved.

2) Federated learning: After the introduction of the vanilla FL strategy FedAvg [8], research attempts to address various practical factors affecting FL performance. The first category of research works deals with computation and networking issues of FL. For instance, the work [16] aims at reducing computation and communication overheads of FL with edge computing clients. The work [17] aims at increasing FL's robustness to straggler clients.

The second category of studies addresses the issues of *global class imbalance* and/or *local class skews*. The former refers to the non-uniform class distribution of all the data in the FL system. The latter refers to that the clients' class distributions are distinct, which is often referred to as the non-IID data problem. Local class skews in general imply global class imbalance. The works [18], [19] only address global class imbalance. Approaches addressing the local class skews problem in FL can be classified as *data-centric*, *system-centric*, and *personalized*. Data-centric approaches [20], [12] apply mechanisms like data sharing between clients, data augmentation to enrich local data diversity, or local data self-balancing. Client clustering [21] is a typical system-centric approach, which selects similar clients to form federation. However, data-centric and system-centric approaches often require excessive communication and storage, making their implementations on resource-constrained devices challenging.

Personalized FL, as surveyed in [22], addresses local class imbalance, often via either adapting a global model trained by FL to the local dataset (e.g., [23]) or applying specific designs of aggregating heterogeneous local models (e.g., [11]). However, the existing personalized FL approaches are in general not designed for MCU-class hardware. This paper exploits a key property of CFC to design a computation- and communication-efficient personalized FL approach that can work on MCU-class hardware.

The third category of studies addresses the FL clients' data distribution skews, i.e., the clients' local datasets form different domains. The work [24] reviews the existing approaches to this issue and applies self-supervised learning to exploit unlabeled local data to manage cross-client domain shifts.

B. Preliminaries on LTC and CFC Neural Networks

Notation convention in this paper is as follows. Take letter x as an example. x denotes a scalar; $\mathcal{X}$ denotes a set; $\mathbf{x}$ denotes a column vector; $\mathbf{x}_{[i]}$ denotes the ith element of $\mathbf{x}$; $\mathbf{X}$ denotes a matrix; $\mathbf{X}_{[i]}$ denotes the ith row of $\mathbf{X}$; $\mathbf{X}[i,j]$ denotes the (i,j)th element of matrix $\mathbf{X}$; $\odot$ denotes Hadamard product.

An LTC of N neurons with M inputs is declared by the following system of linear ODEs:

$$\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = -\boldsymbol{\omega} \odot \mathbf{x}(t) + \mathbf{s}(t), \tag{1}$$

$$\mathbf{s}(t) = \phi\left(\mathbf{x}(t), \mathbf{i}(t); \mathcal{W}\right) \odot \left(\mathbf{a} - \mathbf{x}(t)\right), \tag{2}$$

where $t \in \mathbb{R}$ is continuous time; $\mathbf{x}(t) \in \mathbb{R}^N$ is the hidden state of the neural network; $\boldsymbol{\omega} = [1/\tau_1, 1/\tau_2, \ldots, 1/\tau_N]^\intercal \in \mathbb{R}^N$ with τ_i as a time constant; the ith component of $\mathbf{s}(t) \in \mathbb{R}^N$ represents a non-linear integration of all inputs to the ith neuron; $\mathbf{i}(t) \in \mathbb{R}^M$ is the input signal; $\mathbf{a} \in \mathbb{R}^N$ is constant; $\phi(\mathbf{x}(t), \mathbf{i}(t); \mathcal{W}) \in \mathbb{R}^N$ is a positive, continuous, monotonically increasing and bounded nonlinearity with learnable parameters $\mathcal{W}$. For instance, if the tanh nonlinearity is adopted, the ith component of $\phi(\mathbf{x}(t), \mathbf{i}(t); \mathcal{W})$ can be $\tanh\left(\mathbf{W}_{[i]} \cdot \mathbf{x}(t) + \mathbf{V}_{[i]} \cdot \mathbf{i}(t) + \mathbf{b}_{[i]}\right)$, where the (i,j)th element of the matrix $\mathbf{W} \in \mathbb{R}^{N \times N}$ is the weight of the directional link from the jth neuron to the ith neuron; the (i,k)th element of the matrix $\mathbf{V} \in \mathbb{R}^{N \times M}$ is the weight of the directional link from the kth input to the ith neuron; $\mathbf{b} \in \mathbb{R}^N$ is bias. Thus, $\mathcal{W} = \{\mathbf{W}, \mathbf{V}, \mathbf{b}\}$. When the topology of the directed graph specified by $\mathbf{W}$ contains cycles, the LTC is recurrent. LTC has a basis from biological neural systems. As illustrated by the left part of Fig. 1, for the ith neuron, Eq. (1) characterizes the dynamics of its membrane potential $\mathbf{x}_{[i]}(t)$ given its synaptic current $\mathbf{s}_{[i]}(t)$ [25]; Eq. (2) describes the ith neuron's dendritic integration of its presynaptic sources to generate $\mathbf{s}_{[i]}(t)$ [25]. The model is said to have liquid (i.e., varying) time constant, because the system time constant varies with $\mathbf{x}(t)$ and $\mathbf{i}(t)$. The system time constant is the reciprocal of the coefficient for the term $\mathbf{x}(t)$ in the ODE system specified by Eqs. (1) and (2). The time constant's liquidity improves model's expressivity compared with the models with fixed time constants [4].

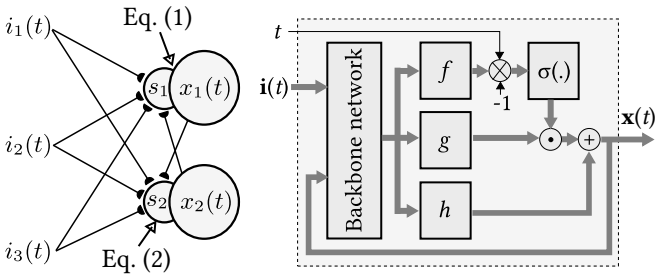


Fig. 1: LTC network (left) and CFC network (right). The illustrated LTC has three presynaptic sources and two neurons. The CFC network has two types of inputs, i.e., a vector $\mathbf{i}(t)$ and a scalar t, and outputs a vector $\mathbf{x}(t)$. Further processing (e.g., by an MLP) can be applied on $\mathbf{x}(t)$ to generate a label.

As discussed in §I, the forwarding and training processes of a CT-NN are slow. A recent work [6] obtains a closed-form approximate solution with known approximation bounds to the scalar version (i.e., $N = 1$) of the ODE system in Eqs. (1) and (2): $x(t) \approx (x(0) - a)\mathrm{e}^{-[\omega + \phi(i(t);\mathcal{W})]t}\phi(-i(t);\mathcal{W}) + a$. The work [6] also describes an algorithm to stack the scalar closed-form approximate solutions for the N dimensions of $\mathbf{x}(t)$ to obtain a vector closed-form approximate solution:

$$\mathbf{x}(t) \approx \boldsymbol{\beta} \odot \mathrm{e}^{-[\boldsymbol{\omega} + \phi(\mathbf{x}(t), \mathbf{i}(t); \mathcal{W})]t} \odot \phi(-\mathbf{x}(t), -\mathbf{i}(t); \mathcal{W}) + \boldsymbol{\alpha}, \tag{3}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^N$ and $\boldsymbol{\beta} \in \mathbb{R}^N$ are constants. To facilitate the implementation of Eq. (3) using modern deep learning toolboxs (e.g., PyTorch), three FNN modules, i.e., $f(\mathbf{x}(t), \mathbf{i}(t); \mathcal{W}^f)$, $g(\mathbf{x}(t), \mathbf{i}(t); \mathcal{W}^g)$, $h(\mathbf{x}(t), \mathbf{i}(t); \mathcal{W}^h)$, are used to represent the following three terms in Eq. (3), respectively: $\boldsymbol{\omega} + \phi(\mathbf{x}(t), \mathbf{i}(t); \mathcal{W})$, $\phi(-\mathbf{x}(t), -\mathbf{i}(t); \mathcal{W})$, and $\boldsymbol{\alpha}$. In the rest of this paper, these three FNNs are called f, g, and h modules. In addition, the exponential decay in Eq. (3) is replaced by a reversed sigmoid. Thus, the CFC is given by

$$\mathbf{x}(t) \approx \sigma\big(-f(\mathbf{x}(t), \mathbf{i}(t); \mathcal{W}^f)t\big) \odot g(\mathbf{x}(t), \mathbf{i}(t); \mathcal{W}^g) + h(\mathbf{x}(t), \mathbf{i}(t); \mathcal{W}_h). \tag{4}$$

where $\sigma(\cdot)$ is a sigmoid satisfying $\sigma(-\infty) = 0$ and $\sigma(0) = 1$. To improve trainability, the three FNN modules f, g, and h share a few feedforward layers called *backbone network*. The right part of Fig. 1 illustrates the structure of CFC.

III. Performance and Properties of CFC

In this section, we propose a reformulated CFC in §III-A to improve CFC's numerical stability, which is important for preventing training instability and performance degradation under quantization on resource-limited IoT devices. The preliminary version of this work [26] presented CFC's better model expressivity within 256 kB SRAM capacity. In §III-B, we focus on conducting benchmark experiments to show the following properties of CFC: low training and inference memory overhead on MCUs, robustness to sampling irregularity, and faster convergence speed under federated learning setting. Lastly, in §III-C, we show a new property of CFC, i.e., its f module's insensitivity to the class distribution of the training data. We also provide the theoretic understanding of the insensitivity. The results in this section form a basis to design FedCFC. The above properties make CFC well-suited for on-device federated learning on resource-constrained IoT devices.

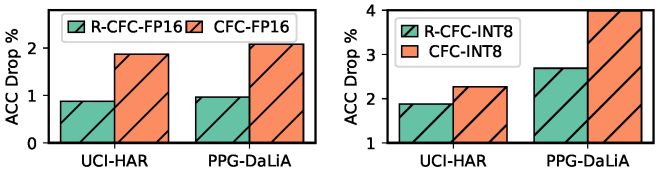
A. Reformulated CFC for Improved Numerical Stability

If a model is numerical stable, small perturbations in input data, parameters, or computational precision do not result in disproportionate large errors. When deploying models on resource-constrained MCUs, quantization that reduces data precision is a technique commonly applied to reduce compute overhead and latency. While CFC's formulation enables efficient computation by avoiding numerical ODE solvers, its numerical stability requires improvement. Specifically, when deploying a single-precision (FP32) CFC model with lower precisions such as half precision (FP16) or 8-bit signed integer (INT8), CFC experiences non-trivial accuracy drops. This issue arises from the multiplicative dependency between the learned decay function $f(\mathbf{x}(t), \mathbf{i}(t); \mathcal{W}^f)$ and time t, which is subsequently passed to the sigmoid $\sigma(\cdot)$. With a lower precision, rounding errors in $f(\mathbf{x}(t), \mathbf{i}(t); \mathcal{W}^f)$ are amplified by t before reaching the sigmoid. For instance, with FP32 precision, suppose $f(\mathbf{x}(t), \mathbf{i}(t)) = 2.7183$ and $t = 20$, then $\sigma(-54.366) \approx 2.96 \times 10^{-24}$. However, with INT8 quantization, $f(\mathbf{x}(t), \mathbf{i}(t))$ is rounded to 3 and we obtain $\sigma(-60) \approx 8.76 \times 10^{-27}$, which is nearly $100\times$ smaller than the original output under the FP32 precision.

To address the issue, we place f function in the denominator to prevent multiplicative amplification of precision errors and ensure smoother dependency on f. Moreover, we replace the sigmoid $\sigma(-f(\mathbf{x}(t), \mathbf{i}(t); \mathcal{W}^f)t)$ with $1 - \exp(-t/(1 + f(\mathbf{x}(t), \mathbf{i}(t); \mathcal{W}^f)))$ to improve numerical stability while retaining smooth growth. The reformulated CFC is given by

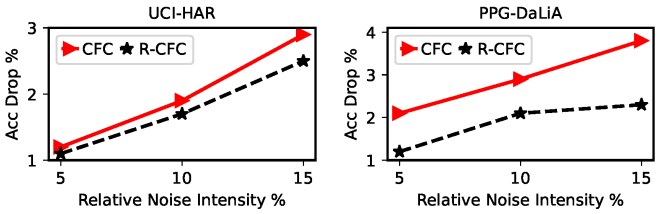
$$\mathbf{x}(t) \approx \left(1 - \mathrm{e}^{-\frac{t}{1+f(\mathbf{x}(t),\mathbf{i}(t);\mathcal{W}^f)}}\right) \odot g(\mathbf{x}(t), \mathbf{i}(t); W^g) + h(\mathbf{x}(t), \mathbf{i}(t); \mathcal{W}^h). \quad (5)$$

Unlike the multiplication operator, the division does not bear the associative property, meaning that small rounding errors are not directly amplified by t but instead influence the denominator in a more controlled manner. This ensures that quantization-induced errors remain bounded and smoothly propagated. Moreover, the function $1 - \frac{1}{\exp(\cdot)}$ is more numerically stable than the sigmoid because it avoids extreme saturation regions and maintains smooth gradient behavior.



(a) Accuracy drop after FP16 quantization. (b) Accuracy drop after INT8 quantization.

Fig. 2: Accuracy drop after quantization.



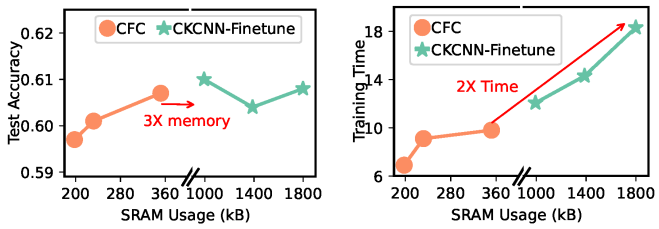
(a) Accuracy drop with noisy input on UCI-HAR dataset. (b) Accuracy drop with noisy input on PPG-DaLiA datset.

Fig. 3: Accuracy drop with noisy input.

We present a benchmark to compare the numerical stability between the reformulated CFC (R-CFC) and the original CFC. We first evaluate the impact of quantization on accuracy, followed by assessing the effect of input noise on accuracy. In the first experiment, we use FP32 precision during training and subsequently use FP16 or INT8 precision during inference. We evaluate on two human activity recognition datasets: the PPG-DaLiA [27] dataset, which predicts activities based on photoplethysmogram signals, and the UCI-HAR [28] dataset, which uses motion acceleration for activity recognition. Fig. 2 shows that when quantized to FP16 precision, R-CFC experiences accuracy drops of 1.6% and 2.7%, which are lower than the 2.3% and 4.0% drops observed in CFC. With INT8 precision, the numerical stability advantage of R-CFC becomes more pronounced, with an average accuracy drop of 3.74% compared with 6.32% for CFC. In the second experiment, we add white noise with a relative intensity ranging from 5% to 15% to the input of the two datasets. Fig. 3 shows that R-CFC exhibits lower accuracy degradation compared with CFC when handling noisy inputs. Additionally, in the absence of noise, the accuracy difference between the two models on both datasets remains within 0.6%. In the rest of this paper, we will not distinguish between R-CFC and CFC, and we will uniformly refer to them as CFC. However, by default, our implementation corresponds to R-CFC.

B. Basic Properties of CFC

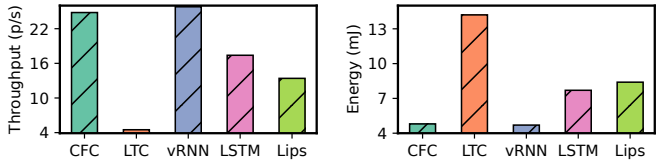
1) On-device training overhead: On-device training capability is essential to continual learning and federated learning. However, on-device training often poses significant challenges for resource-constrained IoT devices. This section provides



(a) Test accuracy vs. on-device training SRAM usage.

(b) On-device training time vs. SRAM usage.

Fig. 4: On-device training overhead of CFC and CKCNN on a Cortex-M7 MCU. We train the entire CFC from scratch but only fine-tune the kernel MLP of CKCNN due to the memory constraint of the MCU.



(a) On-device throughput per second.

(b) On-device energy consumption per inference (mJ)

Fig. 5: Inference throughput and energy usage on Cortex-M7.

a benchmark to show the advantages of CFC for implementing on-device training, through the comparison with the continuous kernel CNN (CKCNN) proposed for sequential data processing [29]. Many time series models cannot be trained on MCUs with 256 KB SRAM because their training processes require much larger memory spaces. To enable a fair comparison between CFC networks and stronger baselines such as CKCNNs, we therefore use the Cortex-M7 core-based STM32H750XB MCU, which is equipped with 2 MB SRAM and a floating-point unit (FPU). All on-device training algorithms are implemented using TinyEngine [15]. Each parameter of CFC or CKCNN is represented by the FP16 format. We use the sequential CIFAR10 dataset, which is constructed from the standard CIFAR10 by flattening each 32×32 RGB image into a one-dimensional sequence of pixels, so that each image is represented as a time series of length 3072. We vary the sizes of the f, g, and h modules to generate various CFC networks. Figs. 4a and 4b show the CFC's test accuracy and training times versus the SRAM usage during the training. Training the entire CKCNN is infeasible on the MCU due to limited SRAM. As a workaround, we measure the overhead of CKCNN-based class-incremental learning. Specifically, we pre-train the CKCNN on a workstation with data in eight out of the ten classes of CIFAR10 and then fine-tune its kernel MLP module on the MCU with data in the remaining two classes. We vary the number of the MLP parameters fine-tuned to obtain the different data points shown in Fig. 4a and 4b for CKCNN. We can see that CFC achieves similar accuracy but with $3\times$ and $2\times$ reduction in SRAM usage and time in on-device training.

TABLE I: Comparison of convergence speed under FL.

Model	#Parameters	#Comm Rounds	#End Acc
CFC	31.9K	**17**	96.1
Neural ODE	31.7K	37	94.3
TCN	30.8K	21	**96.4**
Transformer	30.9K	27	94.2

2) On-device inference overhead: In this section, we evaluate the inference efficiency of CFC compared with RNNs on MCUs with 256 KB SRAM. We measure inference latency and energy consumption on the Cortex-M7 core-based STM32H750XB MCU, with models deployed using FP16 quantization. The models are trained on sequential MNIST. Fig. 5a shows that CFC achieves the highest throughput, exceeding LTC by a factor of $7.2\times$. Fig. 5b presents the energy consumption per inference, measured in millijoules (mJ), where CFC exhibits the lowest energy consumption at approximately 4 mJ, demonstrating its suitability for energy-constrained applications. In contrast, LTC consumes $3\times$ more energy due to the computational overhead of numerical solvers.

3) Processing irregularly sampled data: We consider a scenario where sensor measurements in a trace are collected with irregular sampling intervals. We evaluate on the PhysioNet dataset [30] for heart attack detection using electrocardiograms and the Bosch CNC Machining (CNC) dataset [31] for machine anomaly detection using acoustic signals. The sampling intervals for these datasets vary on the scale of minutes and seconds, with respective variations of 3.7 and 7.1. Our evaluation shows that if we use the average accuracy achieved by the three RNNs as the baseline, CFC outperforms RNNs by 11%, and 3% on the two datasets.

4) Convergence speed under FL: In this section, we evaluate the training convergence speed of the CFC model under a federated learning setting. We adopt FedAvg as the federated algorithm and conduct experiments on the sequential MNIST dataset. For comparison, we include i) a Neural ODE model, where the differential function is parameterized by a 4-layer MLP with 128 hidden units in the middle layers and solved using the RK4 algorithm, ii) a Temporal Convolutional Network (TCN) with three 1D Conv blocks with dilations as $1, 2, 4$ respectivly, 64 channels, and kernel size of 3, and iii) a Transformer baseline consisting of 1 encoder layer with 4 attention heads, and a hidden dimension of 192. These configurations keep the number of the parameters of all the 4 models around 31K. We consider 5 clients, each receiving one-fifth of the full label-balanced training dataset. Training is terminated once the local loss on each client decreases by less than 1% compared with the previous communication round. As shown in Table I, CFC converges in the fewest communication rounds while maintaining competitive accuracy (96.1%), demonstrating its efficiency for federated training. In contrast, the TCN and Transformer require more rounds since they lack an explicit time-continuity mechanism, and the Neural ODE converges the slowest due to the overhead of iterative ODE solvers.

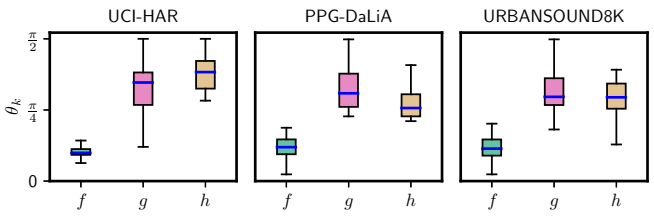


Fig. 6: Principal angle between any two instances of a certain FNN module trained on datasets with class skews.

C. A New Property: f Module's Insensitivity to Label Distribution Shift

This section presents a new property of CFC, i.e., f module's insensitivity with respect to the label distribution shift of the training data. Let $\delta(f) = 1 - \exp\left(-\frac{1}{1+f}\right)$. This function maps the output of f module into the bounded range $(0, 1)$. Thus, the $\delta(f)$ acts as a gate on g module's contribution. When only the label distribution shifts but the feature distribution remains unchanged, the optimal $\delta(f)$ does not vary. Accordingly, since δ is bijective, f itself remains unchanged. In what follows, we first define a formal measure of the insensitivity and then present empirical benchmark results. Finally, we provide a proof of insensitivity in the case of scalar data.

1) Definitions: Denote an L-layer MLP trained with dataset $\mathcal{D}$ by $\pi(\cdot; \mathcal{W}_\mathcal{D})$, where $\mathcal{W}_\mathcal{D} = \{\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_L\}$ and $\mathbf{W}_i \in \mathbb{R}^{n_{i-1} \times n_i}$ is the weight matrix of the connection from the $(i-1)$th layer with n_{i-1} neurons to the ith layer with n_i neurons. Define $\mathbf{\Pi}_\mathcal{D} = \prod_{l=1}^{L} \mathbf{W}_l$. Let $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_P$ denote P partitions of $\mathcal{D}$, i.e., $\bigcup_{p=1}^{P} \mathcal{D}_p = \mathcal{D}$, where $\mathcal{D}_p \cap \mathcal{D}_q$ is unnecessarily empty. Let $\Psi(\mathbf{\Pi}_{\mathcal{D}_p})$ denote the subspace spanned by $\mathbf{\Pi}_{\mathcal{D}_p}$. We define *principal angle* to measure the angular alignment between two subspaces.

Definition 1 (Principal angle)**.** The angle between the subspaces $\Psi(\mathbf{\Pi}_{\mathcal{D}_p})$ and $\Psi(\mathbf{\Pi}_{\mathcal{D}_q})$ is denoted by $\Theta(\mathbf{\Pi}_{\mathcal{D}_p}, \mathbf{\Pi}_{\mathcal{D}_q}) = [\theta_1, \theta_2, \ldots, \theta_k]$, where k is the minimum between the dimensions of the two subspaces; $0 \leq \theta_1 \leq \theta_2 \leq \cdots \leq \theta_k \leq 90°$. The θ_i is recursively defined from $i = 1$ to $i = k$ by $\theta_i = \arccos\left(\frac{|\langle \mathbf{u}_i^*, \mathbf{v}_i^* \rangle|}{\|\mathbf{u}_i^*\| \|\mathbf{v}_i^*\|}\right)$ and $\mathbf{u}_i^*, \mathbf{v}_i^* = \operatorname{argmin}_{\mathbf{u}_i, \mathbf{v}_i} \arccos\left(\frac{|\langle \mathbf{u}_i, \mathbf{v}_i \rangle|}{\|\mathbf{u}_i\| \|\mathbf{v}_i\|}\right)$, where $\mathbf{u}_1$ and $\mathbf{v}_1$ are any column vectors of $\mathbf{\Pi}_{\mathcal{D}_p}$ and $\mathbf{\Pi}_{\mathcal{D}_q}$, respectively; for $i \in [2, k]$, $\mathbf{u}_i \in \Psi(\mathbf{\Pi}_{\mathcal{D}_p})$ and $\mathbf{v}_i \in \Psi(\mathbf{\Pi}_{\mathcal{D}_q})$ satisfy $\mathbf{u}_i \perp \mathbf{u}_j^*$, $\mathbf{v}_i \perp \mathbf{v}_j^*$, $\forall j \in \{1, \ldots, i-1\}$. The angle $\max \Theta(\mathbf{\Pi}_{\mathcal{D}_p}, \mathbf{\Pi}_{\mathcal{D}_q}) = \theta_k$ is called *principal angle*.

Definition 2 (θ-sensitivity)**.** An MLP π is θ-sensitive on dataset partitions $\{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_P\}$ if $\sup_{\forall p,q \in [1,P]} \max \Theta(\mathbf{\Pi}_{\mathcal{D}_p}, \mathbf{\Pi}_{\mathcal{D}_q}) \leq \theta$.

The θ-sensitivity measures the sensitivity of an MLP with respect to the training data provided. Assume two MLPs, π_A and π_B, are θ_A-sensitive and θ_B-sensitive, respectively, on the same training dataset. If $\theta_A \geq \theta_B$, we say π_A is more sensitive than π_B with respect to the training data provided.

2) Insensitivity of f module to label distribution shift: We compare the sensitivity of CFC's three FNN modules, f, g, and h, as well as other neural networks to label distribution shifts. Specifically, we compare the principal angles of the f, g, and h modules concerned in the condition of Definition 2. In particular, we focus on the sensitivity with respect to the label distribution skew of training data. Specifically, the sets of classes contained in the partitions $\mathcal{D}_1, \ldots, \mathcal{D}_P$ are distinct. We use the following three datasets: UCI-HAR [28], PPG-DaLiA [27], and URBANSOUND8K [32]. UCI-HAR and PPG-DaLiA have been introduced in §III-A. URBANSOUND8K includes acoustic traces collected from different locations in urban areas. We use the traces to infer their collection locations. From a dataset with K classes, we create C_2^K partitions, where each partition contains data in two classes. Then, we measure the principal angle between two instances of a certain FNN module trained on any two partitions. This generates $C_2^{C_2^K}$ principal angles. Fig. 6 shows the distribution of such $C_2^{C_2^K}$ principal angles of each of f, g, and h. We can see that the f module has the smallest principal angles in any dataset. This suggests that f is less sensitive than g and h. Empirically, the f modules with ReLU activation for the three datasets are 24°-, 33°-, and 36°-sensitive, respectively. The conference paper of this work [26] also compared the sensitivity of CFC's f module and other neural networks including CKCNN, vRNN, and GRU. It also showed that the f's insensitivity does not generalize to the cross-dataset cases.

3) Theoretic understanding of module f's insensitivity:

Proposition 1 (Bounded Gradient Sensitivity to Label Shift)**.** *Consider two labeled datasets $\mathcal{D}_1$ and $\mathcal{D}_2$ with data distributions of $p_1(i(t), y) = p_1(y)p_1(i(t)|y)$ and $p_2(i(t), y) = p_2(y)p_2(i(t)|y)$, respectively, where $i(t)|_{t=0}^{T}$ is the scalar time series data for a time duration of T, $y \in [0, 1]$ is the corresponding label after min-max normalization. The datasets have different label distributions (i.e., $p_1(y) \neq p_2(y)$, $\forall y$) and bounded feature distribution shifts (i.e., $|p_1(i(t)|y) - p_2(i(t)|y)| \leq \tau$, $\forall y$). Denoting by $\mathcal{L}_i$ the loss of the CFC model trained on $\mathcal{D}_i$ and by $\nabla_{\mathcal{W}^f} \mathcal{L}_i$ the gradient of $\mathcal{L}_i$ with respect to the f module's parameters $\mathcal{W}^f$, we have $|\nabla_{\mathcal{W}^f} \mathcal{L}_1 - \nabla_{\mathcal{W}^f} \mathcal{L}_2| \leq \frac{\tau L_H}{2} + \frac{8 L_f L_\mathcal{L} \mathrm{e}^{-2}}{T}$, where L_H,L_f,$L_\mathcal{L}$ are positive constants.*

Proof. In this proof, we omit the subscript i when the derivation is agnostic to i. The formal representation of the gradient is $\nabla_{\mathcal{W}^f} \mathcal{L} = \mathbb{E}_{(i(t),y) \sim p(i(t),y)} [\nabla_{\mathcal{W}^f} \mathcal{L}(i(t), y)]$. We have $\nabla_{\mathcal{W}^f} \exp\left(-\frac{t}{1+f}\right) = \frac{t \exp(-t/(1+f))}{(1+f)^2} \nabla_{\mathcal{W}^f} f$. By applying the chain rule and decomposing the expectation, we have $\nabla_{\mathcal{W}^f} \mathcal{L} = \sum_y p(y) \cdot \mathcal{C}_i$, where $\mathcal{C}_i = \mathbb{E}_{i(t) \sim p_i(i(t)|y)} \left[\frac{t \mathrm{e}^{\frac{-t}{1+f}}}{(1+f)^2} \nabla_{\mathcal{W}^f} f \cdot \nabla_{i(t)} \mathcal{L}_i\right]$.

In the following, we prove that $|\mathcal{C}_1 - \mathcal{C}_2|$ is bounded. Let $H(i(t)) = \frac{t \mathrm{e}^{\frac{-t}{1+f}}}{(1+f)^2} \nabla_{\mathcal{W}^f} f \cdot \nabla_{i(t)} \mathcal{L}$. As both $\frac{t \mathrm{e}^{\frac{-t}{1+f}}}{(1+f)^2}$ and $\nabla_{i(t)} \mathcal{L}$ are continuously differentiable against $i(t)$, the function $H(i(t))$ is Lipschitz continuous [33]. Denote by L_H the Lipschitz constant of $H(i(t))$. By applying the Kantorovich-Rubinstein Duality theorem [34], we have $|\mathcal{C}_1 - \mathcal{C}_2| \leq$

$L_H W_1(p_1(i(t)|y), p_2(i(t)|y))$, where W_1 is the Wasserstein-1 distance. Since $\forall y \in [0,1], |p_1(i(t)|y) - p_2(i(t)|y)| \leq \tau$, we have $W_1(p_1(i(t)|y), p_2(i(t)|y)) \leq \frac{\tau}{2}$ and $|\mathcal{C}_1 - \mathcal{C}_2| \leq \frac{\tau L_H}{2}$.

In what follows, we prove that $|\mathcal{C}_i|$ is bounded. Because the f and $\mathcal{L}$ are Lipschitz-continuous, we have $\sup|\nabla_{\mathcal{W}^f} f| = L_f$ and $\sup|\nabla_{i(t)}\mathcal{L}| = L_{\mathcal{L}}$, where L_f and $L_{\mathcal{L}}$ are the Lipschitz constants of f and $\mathcal{L}$, respectively. Define function $G(f) = \frac{te^{\frac{-t}{1+f}}}{(1+f)^2}$. As $f > -1$ which holds for most activation functions including ReLU, tanh, and sigmoids, we have $\sup|G(f)| = \sup G(f) = \sup G(\frac{t-2}{2}) = \frac{4e^{-2}}{t}$. Therefore, $|\mathcal{C}_i| \leq |G(f)| \cdot |\nabla_{\mathcal{W}^f} f| \cdot |\nabla_{i(t)}\mathcal{L}| \leq \frac{4L_f L_{\mathcal{L}} e^{-2}}{t}$. Thus, we have:

$$
\begin{aligned}
|\nabla_{\mathcal{W}^f}\mathcal{L}_1 - \nabla_{\mathcal{W}^f}\mathcal{L}_2| &= \left|\sum_y p_1(y)\mathcal{C}_1 - p_2(y)\mathcal{C}_2\right| \\
&\leq \sum_y (|p_1(y) - p_2(y)| \cdot |\mathcal{C}_1| + p_2(y) \cdot |\mathcal{C}_1 - \mathcal{C}_2|) \\
&\leq \frac{\tau L_H}{2} + \frac{4L_f L_{\mathcal{L}} e^{-2}}{t} \sum_y |p_1(y) - p_2(y)| \\
&\leq \frac{\tau L_H}{2} + \frac{8L_f L_{\mathcal{L}} e^{-2}}{t}.
\end{aligned}
$$

After training completes, $t = T$. Therefore, we have $|\nabla_{\mathcal{W}^f}\mathcal{L}_1 - \nabla_{\mathcal{W}^f}\mathcal{L}_2| \leq \frac{\tau L_H}{2} + \frac{8L_f L_{\mathcal{L}} e^{-2}}{T}$. □

The above proposition implies that if $\tau = 0$ (i.e., no feature distribution shift) and $T \to \infty$ (i.e., a sufficiently large dataset), $|\nabla_{\mathcal{W}^f}\mathcal{L}_1 - \nabla_{\mathcal{W}^f}\mathcal{L}_2| = 0$. Since the gradient difference of the f-module is provably bounded by the label distribution divergence, the variation of f's weights across clients is bounded. This establishes the insensitivity of the f-module to label drift. The above analysis provides insight into understanding the insensitivity of the f module against the label distribution shift.

IV. CFC-Based Federated Learning Network

A. Motivation

The basic properties of CFC shown in §III-B suggest CFC's advantages on single IoT devices. A natural question next is whether we can build an efficient FL network based on CFC for an IoT network. FL well matches the distributed nature of IoT networks and possesses a key advantage of retaining the training data at IoT devices, which has a privacy-preserving implication. A well-designed FL system can learn more versatile models, as it uses wider training data. When the training data at each client is limited, FL can increase the model accuracy substantially compared with the case where the clients train their models independently.

However, the *local class skews* (i.e., non-IID data) problem [22] has challenged efficient FL designs. The problem is formally stated as follows. For a set of clients $\mathcal{C}$, let $\mathcal{D}_c = \{(\mathbf{X}_c, Y_c)\}$ denote client c's local dataset, where $c \in \mathcal{C}$, $\mathbf{X}_c$ is a data sample and Y_c is the corresponding label. The $\mathcal{D}_c$ can be viewed as a partition of the global dataset $\mathcal{D} = \bigcup_{c\in\mathcal{C}} \mathcal{D}_c$. In this paper, we assume that the conditional probability distribution $P(\mathbf{X}_c|Y_c = y)$ is identical across all clients. That is, the clients' local datasets have no domain shifts. The local class skews problem refers to that the prior probability distribution $P(Y_c)$ varies with c. It impedes FL's capability to let each local model converge to the desired global optimal model that minimizes the loss function on $\mathcal{D}$. This is because the losses computed by the clients with their local datasets are distinct and therefore the local model updates can be hardly harmonized. If vanilla FL strategy FedAvg is applied, oscillations in the global model updates can be observed.

While the existing studies have proposed various FL strategies to deal with the local class skews as discussed in §II-A, the CFC f module's insensitivity with respect to $P(Y_c)$ offers an opportunity to design a simple yet effective FL strategy. Specifically, the f module captures the task-wide pattern common across the non-IID dataset partitions, while the g, h modules capture the partition-specific patterns. This separation inspires the basic idea of the proposed FedCFC, i.e., we federate the f modules learned by the clients, while leaving other parts of CFC unfederated. This configuration offers two advantages. First, the federated learning for f can improve its versatility while not facing significant oscillations supposedly, since the local updates to f are likely harmonized owing to f's insensitivity. Second, leaving the g and h unfederated allows the local models to be personalized. Note that applying existing non-IID FL strategies on g and h might bring some further accuracy improvement. However, they also introduce more computation and communication overheads, as shown in §V.

B. Overview of the Federated Learning Network

A straw man solution is to apply FedAvg on the f modules, which is called FedCFC-fAvg in this paper. In each communication round of FedCFC-fAvg, the server disseminates the global f module. Then, each client sends back the update. Although the local f modules across the clients with non-IID data are nearly identical after the convergence of FedCFC-fAvg, they traverse different trajectories during the FL process. In this paper, we propose a more efficient FL strategy called FedCFC-fBind, where "fBind" refers to the strategy that explicitly binds the evolution trajectories of the clients' f modules. Its efficiency is from two aspects. First, it explicitly preserves the insensitivity of f throughout the FL process. This preservation aims at harmonizing the local updates to f, speeding up the convergence. Second, different from FedCFC-fAvg that exchanges the f module between the server and client, FedCFC-fBind only exchanges the gradient of the f module's last layer, reducing communication overhead. Its formalization and workflow are as follows.

Let $\mathcal{W}_c^f$, $\mathcal{W}_c^g$, and $\mathcal{W}_c^h$ denote the parameters of the f, g, and h modules of client c's CFC. Denote $\mathcal{W}_c = \{\mathcal{W}_c^f, \mathcal{W}_c^g, \mathcal{W}_c^h\}$. Let $L(\mathcal{D}_c, \mathcal{W}_c)$ denote the cross-entropy loss at client c. Each round of FedCFC-fBind aims at solving

$$
\min_{\mathcal{W}_c, \forall c\in\mathcal{C}} \sum_{\forall c\in\mathcal{C}} L(\mathcal{D}_c, \mathcal{W}_c) \text{ s.t. } \max \Theta(\mathbf{\Pi}^f_{\mathcal{D}_p}, \mathbf{\Pi}^f_{\mathcal{D}_q}) \leq \theta, \forall p, q \in \mathcal{C}, \tag{6}
$$

where $\Theta(\mathbf{\Pi}^f_{\mathcal{D}_p}, \mathbf{\Pi}^f_{\mathcal{D}_q})$ is the principal angle between the f modules of any two clients p and q. The constraint in Eq. (6) preserves the insensitivity of the f module. Although the θ in

Eq. (6) can adopt a setting according to the prior knowledge about the f module's θ-volatility, it will not be needed in a Lagrangian relaxation of Eq. (6) solved by FedCFC-fBind, which will be presented in §IV-C shortly. In each communication round of FedCFC-fBind, the clients and parameter server perform the following actions.

Client step: A client applies stochastic gradient descent (SGD) to solve an unconstrained optimization problem relaxed from Eq. (6). The relaxation uses regularization based on a common update direction (CUD) and a common update magnitude (CUM) received from the parameter server regarding the f. The local model performs E local updates in each round. This is called *bound local updater*. Then, the client reports the local update direction (LUD) to the parameter server.

Server step: Based on the LUDs received from the clients, the server applies an optimization-based geometric aggregation algorithm called *bound updates aggregator* to generate the next CUD and CUM, and disseminates them to the clients.

Through iterating the above steps, the clients' f modules keep aligned throughout the FL process. The next subsections present the details of the client's and server's steps.

C. Client's Bound Local Updater

Client c applies SGD to update its f, aiming to minimize the overall loss $L_c = l_t + \lambda_d R_d + \lambda_m R_m + \lambda_a R_a$, where l_t is the loss of the task (e.g., cross-entropy loss of classification task); R_d, R_m, and R_a are three regularization terms presented below; the lambdas are hyperparameter weights. The above unconstrained optimization encompasses the Lagrangian relaxation to Eq. (6). At the end of local update, client c transmits the gradient of L_c with respect to the weights in the last layer of f module, denoted by $\nabla_f L_c$, as the LUD to the server.

The *directional* regularization term R_d is the cosine distance between $\nabla_f l_t$ and CUD, i.e., $R_d = 1 - \frac{\nabla_f l_t \cdot \text{CUD}}{\|\nabla_f l_t\|_2 \|\text{CUD}\|_2}$, where $\nabla_f l_t$ is the gradient of l_t with respect to the weights in the last layer of module f and the CUD is disseminated by the parameter server in the last round. Thus, R_d encourages the directional alignment between the local gradient update and the CUD suggested by the server.

The *magnitudinous* regularization term R_m is given by $R_m = \max(0, \|\nabla_f l_t\|_2 - \text{CUM})$, where the CUM is disseminated by the parameter server in the last round. Thus, R_m encourages that the magnitude of the local gradient update remains within CUM.

The R_d and R_m together bind all clients' local gradient updates. However, they may lead to widespread inactive neurons in the f module. This can impede the convergence of the FL process. The *active* regularization term R_a is applied to increase the activity of neurons. It is given by $R_a = \Upsilon(\mathcal{W}_c^f) - \upsilon$, where υ is a desired activity threshold and $\Upsilon(\mathcal{W}_c^f)$ is the mean activation of the neurons in the module f. Specifically, $\Upsilon(\mathcal{W}) = \min_{i=1}^{L} \max_{j=1}^{n_i} o_{ij}$, where L is the number of layers of the MLP f, n_i is the width of the ith layer, and o_{ij} represents the output of the jth neuron of the ith layer. Note that υ is a hyperparameter.

The right part of Fig. 7 illustrates the local updates at two clients in two communication rounds, where the two dotted red arrows are two CUDs and the two dashed circles are two CUMs received from the server. Solid arrow and dash arrow represent the unregularized and regularized local updates.

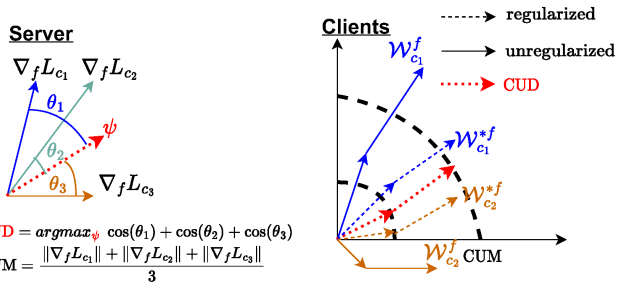


Fig. 7: Illustrations of the server and client steps. *Server*: Given gradients from three clients, the server sets CUD to be the direction ψ that maximizes the sum of cosine similarities with the three client gradients, and CUM to be the average magnitude of the three gradients. *Clients*: The drawing illustrates the trajectories of two clients in two communication rounds. The regularization penalizes direction deviation from CUD and magnitude overstepping from CUM.

D. Server's Bound Updates Aggregator

The aggregator running at the parameter server determines CUM and CUD based on the LUDs received from all clients, i.e., $\{\nabla_f L_c | c \in \mathcal{C}\}$. The aggregator aims to harmonize the clients' local updates. The CUM is determined by $\text{CUM} = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \|\nabla_f L_c\|_2$. We aim to find a CUD to maximize the overall cosine similarity between the CUD and the LUDs, i.e., $\text{CUD} = \text{argmax}_{\boldsymbol{\psi}} \sum_{c \in \mathcal{C}} \frac{\nabla_f L_c \cdot \boldsymbol{\psi}}{\|\nabla_f L_c\|_2 \|\boldsymbol{\psi}\|_2}$. This is illustrated by the left part of Fig. 7. As it is non-convex optimization, gradient-based methods can be easily stuck at local optimums.

We approach the above problem in a Hilbert space denoted by $\mathbb{H}$. Specifically, we minimize the distance between the CUD and the centroid of the LUDs in $\mathbb{H}$, which is convex. Thus, the key is to find the mapping function from the gradient space (denoted by $\mathbb{G}$) to $\mathbb{H}$. In what follows, we present a proposition, which is an application of the Moore Aronszajn Theorem [35] to our context and will be used to develop the solution to the original CUD determination problem.

Proposition 2. *For any mapping $\Gamma : \mathbb{G} \to \mathbb{H}$, there exists a kernel function $\kappa(\cdot, \cdot)$ defined on domain $\mathbb{G} \times \mathbb{G}$ such that the Hilbert-space distance* $\left\| \frac{1}{|\mathcal{C}|} \sum_{\forall c \in \mathcal{C}} \Gamma(\nabla_f L_c) - \Gamma(\boldsymbol{\psi}) \right\|_{\mathbb{H}}^2 = \text{tr}(\mathbf{AB})$, *where* $\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \in \mathbb{R}^{|\mathcal{C}| \times |\mathcal{C}|} & \mathbf{A}_2 \in \mathbb{R}^{|\mathcal{C}| \times 1} \\ \mathbf{A}_3 \in \mathbb{R}^{1 \times |\mathcal{C}|} & \kappa(\boldsymbol{\psi}, \boldsymbol{\psi}) \end{bmatrix}$, $\mathbf{B} \in \mathbb{R}^{(|\mathcal{C}|+1) \times (|\mathcal{C}|+1)}$, $\mathbf{A}_1[i,j] = \kappa(\nabla_f L_i, \nabla_f L_j)$, $\mathbf{A}_2[i,1] = \kappa(\nabla_f L_i, \boldsymbol{\psi})$, $\mathbf{A}_3[1,j] = \kappa(\boldsymbol{\psi}, \nabla_f L_j)$,

$$\mathbf{B}[i,j] = \begin{cases} \frac{1}{|\mathcal{C}|^2} & \text{if } i \le |\mathcal{C}|, j \le |\mathcal{C}|; \\ 1 & \text{if } i > |\mathcal{C}|, j > |\mathcal{C}|; \\ -\frac{2}{|\mathcal{C}|} & \text{otherwise.} \end{cases}$$

Proof. Let $\langle \cdot, \cdot \rangle_{\mathbb{H}}$ denote the inner product operation defining $\mathbb{H}$. The Moore Aronszajn theorem [35] states that $\forall (\mathbf{x}, \mathbf{x}') \in$

Algorithm 1: FedCFC-fBind

Data: Local datasets $\mathcal{D}_c$, $\forall c \in \mathcal{C}$
Result: Local models $\mathcal{W}_c = \{\mathcal{W}_c^f, \mathcal{W}_c^g, \mathcal{W}_c^h\}$, $\forall c \in \mathcal{C}$
Configuration: Learning rate η; regularization weights λ_d, λ_m, λ_a; neuron activation threshold υ
while *not converged* **do**
  server: disseminates CUD and CUM to clients
  for *each client* c *in parallel* **do**
    Compute loss l_t and gradients $\nabla_f l_t$, $\nabla_g l_t$, $\nabla_h l_t$
    $\mathcal{W}_c^g \leftarrow \mathcal{W}_c^g - \eta \dot{\nabla}_g l_t, \quad \mathcal{W}_c^h \leftarrow \mathcal{W}_c^h - \eta \dot{\nabla}_h l_t$
    $R_d \leftarrow 1 - \frac{\nabla_f l_t \cdot \text{CUD}}{\|\nabla_f^f l_t\|_2 \|\text{CUD}\|_2}$,
    $R_a \leftarrow \Upsilon(\mathcal{W}_c^f) - \upsilon$
    $R_m \leftarrow \max(0, \|\nabla_f l_t\|_2 - \text{CUM})$
    $L_c \leftarrow l_t + \lambda_d R_d + \lambda_m R_m + \lambda_a R_a$ and compute $\nabla_f L_c$
    $\mathcal{W}_c^f \leftarrow \mathcal{W}_c^f - \eta \dot{\nabla}_f L_c$
    Upload $\nabla_f L_c$ to server
  server: $\text{CUM} \leftarrow \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \|\nabla_f L_c\|_2$, compute CUD

$\mathbb{G} \times \mathbb{G}$, $\forall \Gamma$, $\exists \kappa$, such that $\kappa(\mathbf{x}, \mathbf{x}') = \langle \Gamma(\mathbf{x}), \Gamma(\mathbf{x}') \rangle_{\mathbb{H}}$. Therefore, we have $\left\| \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \Gamma(\nabla_f L_c) - \Gamma(\boldsymbol{\psi}) \right\|_{\mathbb{H}}^2 = \frac{1}{|\mathcal{C}|^2} \sum_{\forall p,q \in \mathcal{C}} \langle \Gamma(\nabla_f L_p), \Gamma(\nabla_f L_q) \rangle + \langle \Gamma(\boldsymbol{\psi}), \Gamma(\boldsymbol{\psi}) \rangle - \frac{2}{|\mathcal{C}|} \sum_{\forall c \in \mathcal{C}} \langle \Gamma(\nabla_f L_c), \Gamma(\boldsymbol{\psi}) \rangle \overset{(*)}{=} \frac{1}{|\mathcal{C}|^2} \sum_{\forall p,q \in \mathcal{C}} \kappa(\nabla_f L_p, \nabla_f L_q) + \kappa(\boldsymbol{\psi}, \boldsymbol{\psi}) - \frac{2}{|\mathcal{C}|} \sum_{\forall c \in \mathcal{C}} \kappa(\nabla_f L_c, \boldsymbol{\psi}) \overset{(\dagger)}{=} \text{tr}(\mathbf{AB})$, where the step marked by (*) follows the Moore Aronszajn theorem. □

Now, we discuss how Proposition 2 is related to the original objective of maximizing the overall cosine similarity between the CUD and the LUDs. If $\kappa(\cdot, \cdot)$ in Proposition 2 satisfies that $\kappa(\mathbf{x}, \mathbf{x}')$ is a constant, from the step marked by (†), we have $\text{argmax}_{\boldsymbol{\psi}} \sum_{\forall c \in \mathcal{C}} \kappa(\nabla_f L_c, \boldsymbol{\psi}) = \text{argmin}_{\boldsymbol{\psi}} \text{tr}(\mathbf{AB})$, where the left-hand side of the above equation is a more general form of the original objective. Note that the Gaussian kernel is an example $\kappa(\cdot, \cdot)$ meeting the constant condition. Although, the objective function $\text{argmin}_{\boldsymbol{\psi}} \text{tr}(\mathbf{AB})$ for any given $\kappa(\cdot, \cdot)$ that satisfies the constant condition is still convex optimization problem, searching for an appropriate kernel function is conducted within a large, often infinite-dimensional space.

We follow the relax-then-check strategy presented below to avoid the direct search for the kernel function. The main idea is that, instead of finding the kernel itself, we focus on directly determining the outcome that this kernel would produce.

Relax: Ignore $\mathbf{A}$'s internal structure and solve semi-definite programming: $\{\mathbf{X}^*\} = \text{argmin}_{\mathbf{X} \in \mathbb{R}^{(|\mathcal{C}|+1) \times (|\mathcal{C}|+1)}} \text{tr}(\mathbf{XB})$. The solution $\{\mathbf{X}^*\}$ can be an infinite set.

Check: Check a sufficient number of $\mathbf{X}^*$ within the latency requirement imposed on the aggregator. For each $\mathbf{X}^*$, apply the kernel principal component analysis [36] to determine the $\boldsymbol{\psi}^*$ by $\boldsymbol{\psi}^*_{[i]} = \sum_{j=1}^{|\mathcal{C}|+1} \mathbf{v}_{i[j]} \mathbf{X}^*[|\mathcal{C}|+1, j]$, where $\mathbf{v}_i$ is the eigenvector of $\mathbf{X}^*$ corresponding to the ith largest eigenvalue. The $\boldsymbol{\psi}^*$ that best fits $\mathbf{A}$ with its internal structure defined by the cosine similarity kernel is yielded as the CUD.

E. Entire FedCFC-fBind Process

Algorithm 1 shows the entire FedCFC-fBind process, including the clients' local updates for f, g, h, and the parameter server's aggregation.

V. Performance Evaluation

A. Evaluation Methodology and Settings

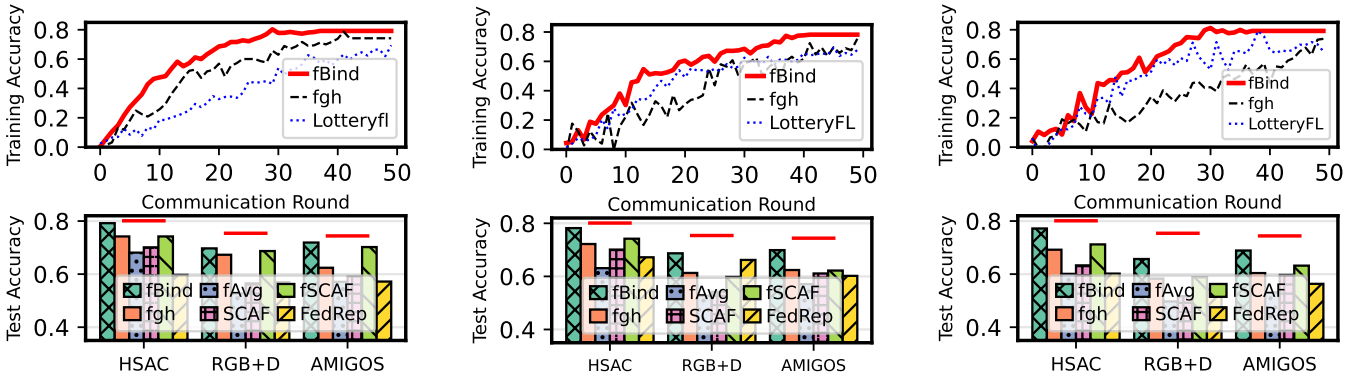
Our evaluation is conducted on the following three datasets:

- **HASC-PAC2016** [37] consists of 19,172 data samples collected from an inertial measurement unit (IMU) worn on the wrist for six-class human activity recognition. Each recording lasts 300 seconds and corresponds to one of six activity classes. For model training, each trace is segmented into 3-second input samples.
- **NTU RGB+D** [38] contains 3D skeletal motion data from 25 joints, covering 60 different human activities. Each input sample comprises a sequence of 3,600 readings across 25 joints, providing detailed motion information for activity recognition. Out of 60 activity labels, 24 labels are used in our experiments.
- **AMIGOS** [39] includes electroencephalogram (EEG) and electrocardiogram (ECG) recordings from 40 participants, annotated with six emotion recognition labels. A sliding window approach with a 1-second window size and 0.2-second overlap is used to segment traces into input samples.

Our evaluation replicates two practical challenges encountered in real-world FL deployments. The first challenge is label distribution shift. We consider two scenarios: class heterogeneity and local class skew. In the first scenario, each client's dataset contains only a small subset of the total classes. In the second scenario, all clients have data from every class, but the label distributions vary significantly across clients. This paper presents results of the first scenario. Please refer to our conference paper [26] for results of the second scenario. The second challenge is scalability with the number of clients, where many MCUs participate in FL but contribute only small amounts of data due to their low data collection rates. Unlike traditional FL settings where each client holds a substantial local dataset, this paradigm involves a vast number of MCUs, each providing limited but diverse data samples.

For each recreated situation, we compare the following FedCFC variants and two existing non-IID FL approaches:

- **FedCFC-fBind** is our main proposal described in §IV.
- **FedCFC-fAvg** uses FedAvg [8] to aggregate f modules.
- **FedCFC-fSCAFFOLD** uses SCAFFOLD [10] to aggregate local f modules. SCAFFOLD corrects local updates using variance reduction to align local models with the global model, thereby mitigating the effects of non-IID data.
- **FedCFC-fgh** uses fBind to federate local f modules and SCAFFOLD to federate the local g and h modules.
- **FedCFC-SCAFFOLD** uses SCAFFOLD to aggregate the local f, g, h modules.



(a) $\gamma = 1$ (IID) (b) $\gamma = 0.5$ (c) $\gamma = 0$ (disjoint)

Fig. 8: Performance of FL approaches under three degrees of class heterogeneity γ. Top row: training accuracy vs. communication rounds; Bottom row: test accuracy after FL converges. The "FedCFC-" prefix is omitted in legends.

- **FedCFC-FedRep** uses FedRep [40] to aggregate the local f, g, h modules. FedRep is a federated learning algorithm that trains a shared global representation across clients while keeping client-specific heads local for personalization.
- **LotteryFL** [11] is personalized FL that only federates a sub-model. It is designed to address non-IID data.
- **BalanceFL** [12] is an FL approach that applies a local self-balancing technique to deal with non-IID data issue.

All the aforementioned FL approaches are implemented using PyTorch. For all FedCFC variants, the CFC model comprises a single-layer backbone along with three two-layer MLPs serving as the f, g, and h modules. It is important to note that the on-device FedCFC implementation, discussed in §VI, does not utilize PyTorch. In both implementations, the batch size is set to one to ensure consistency and to facilitate on-device training. The hyperparameters for our FedCFC-fBind implementation are configured as follows: $\lambda_d = 0.6$, $\lambda_m = 0.2$, $\lambda_a = 0.2$, $\upsilon = 1.59$, and $\eta = 0.01$. Please refer to our conference paper [26] for the sensitivity analysis on the three regularization coefficients λ_d, λ_m, and λ_a.

B. Evaluation with Class Heterogeneity

In this section, we evaluate the performance of FedCFC-fBind under class heterogeneity, where different clients are assigned disjoint subsets of labels to simulate label distribution shifts among clients in federated learning. In this set of experiments, the dataset partition owned by each client only covers a subset of the classes of the entire dataset. When assigning the data samples to a total of 12 clients, we control the degree of class heterogeneity among the clients, denoted by γ. Specifically, $\gamma = |\mathcal{D}_p \cap \mathcal{D}_q|/|\mathcal{D}_p|$. We adopt three γ settings: 1) $\gamma = 1$ means there is no label distribution shifts, in which each client has data in all classes; 2) $\gamma = 0.5$ means a moderate label distribution shift case, in which any two clients' dataset partitions have 50% overlap in terms of classes; 3) $\gamma = 0$ means the most severe label distribution shift case, in which the clients' dataset partitions are disjoint in terms of classes. BalanceFL is skipped in this subsection, because it is inapplicable in the $\gamma = 0.5$ and $\gamma = 0$ cases. In each round of the experiment, we vary the classes each client obtains while fixing the degree of class heterogeneity γ.

The three columns of Fig. 8 show the results under the three γ settings. The top row shows the training accuracy versus the number of communication rounds during the FL process. We only plot the resulons of three approaches for clarity of the figure. The smoothness of the trajectories is negatively affected by the class heterogeneity among the clients. Nevertheless, the trajectory of FedCFC-fBind is smoother than others. The bottom row shows the test accuracy after FL converges. Here, we assume that the convergence is achieved when all clients' validation accuracy improvement is less than 0.05% of the maximum validation accuracy for the first time. In Fig. 8, the horizontal red lines represent the test accuracy when CFC is trained in the centralized manner. It shows that when the data distribution is IID, the difference between the accuracy of centralized training and the accuracy of FedCFC-fBind is within 6%. In addition, FedCFC-fBind achieves higher test accuracy on the three datasets, compared with all FedCFC variants. FedCFC-fBind is also more robust to the class heterogeneity. For instance, on the AMIGOS dataset, FedCFC-fBind has a test accuracy drop of 3.04% when γ decreases from 1 to 0. Additionally, we increase the number of client to 50 and conduct the same experiment on the NTU RGB-D dataset. When γ is 0, 0.5, and 1, the average test accuracy of FedCFC-fBind only shows a drop of 3.9%, 3.19%, and 2.70%, respectively. Among all compared approaches, FedCFC-fAvg gives the lowest test accuracy, because FedAvg is a vanilla FL strategy without non-IID considerations. This also suggests that, specific designs in the FL strategy are needed to exploit the f's insensitivity. FedCFC-FedRep achieves similar accuracy compared with FedCFC-fAvg, because it may be difficult to train a common representation for each of the f,g,h modules among the clients. Compared with FedCFC-fBind, FedCFC-fgh does not show advantages although it additionally applies SCAFFOLD to federate g and h. This is due to the inherent difficulty in federating heterogeneous models. Note that the FedCFC-SCAFFOLD's accuracy is lower than the FedCFC-fSCAFFOLD's accuracy. This is because the former federates

TABLE II: Convergence speed and communication overhead under three settings for the degree of class heterogeneity γ.

Approach	Dataset	γ = 1 (IID)		γ=0.5		γ=0 (disjoint)	
		Comm. rounds	Comm. vol. (kB)	Comm. rounds	Comm. vol. (kB)	Comm. rounds	Comm. vol. (kB)
FedCFC-fBind	HSAC	39	102.5	45	120.2	37	91.2
	NTU RGB+D	43	119.7	44	100.6	49	177.1
	AMIGOS	29	68.1	36	89.8	35	83.2
FedCFC-fgh	HSAC	42	3992.5	71	7000.7	87	9772.1
	NTU RGB+D	64	6332.1	89	10023.2	91	13622.1
	AMIGOS	41	3779.1	62	6017.4	57	5729.2
FedCFC-FedRep	HSAC	35	98.1	59	123.1	50	98.3
	NTU RGB+D	63	147.3	52	110.2	62	192.6
	AMIGOS	43	67.3	58	82.8	58	103.4
FedCFC-SCAFFOLD	HSAC	41	3477.1	58	6213.3	63	8977.1
	NTU RGB+D	57	7051.1	74	7981.6	66	9973.6
	AMIGOS	37	3408.3	48	7321.4	59	6922.2
FedCFC-fAvg	HSAC	47	4517.1	59	8788.2	99+	19379.7
	NTU RGB+D	57	8351.1	77	11311.7	99+	24321.7
	AMIGOS	39	6147.3	45	7322.8	67	9847.2
FedCFC-fSCAFFOLD	HSAC	39	3318.7	49	3988.2	57	5123.2
	NTU RGB+D	51	4122.7	54	4796.4	69	6125.0
	AMIGOS	33	2788.4	36	3072.8	48	4946.4
LotteryFL	HSAC	57	1111.5	61	1188.3	81	1569.7
	NTU RGB+D	72	1287.3	70	1301.7	91	1688.0
	AMIGOS	49	984.9	46	912.6	76	1463.2

all f, g, h modules, while the latter only federates f modules. The g and h modules are kept local to each client, which allows them to better adapt to the local data distribution.

Table II shows the number of communication rounds to achieve convergence and the associated per-client communication volume. We can see that FedCFC-fBind converges faster. In summary, FedCFC-fBind speeds up the convergence by 16.4%, 34.8%, and 45.5% with respect to the average of other three FedCFC variants on all datasets, when $\gamma = 1$, $\gamma = 0.5$, and $\gamma = 0$, respectively. The speed-up ratios with respect to LotteryFL are 35.5%, 33.4%, and 50.8%. As FedCFC-fBind only transmits f's last layer gradient, it is communication-efficient. FedCFC-fBind reduces per-client communication volume by 51×, 72×, and 96× with respect to the average of the other three FedCFC variants on all datasets, when $\gamma = 1$, $\gamma = 0.5$, and $\gamma = 0$, respectively. The reduction ratios with respect to LotteryFL are 11×, 12×, and 14×. The lower communication overhead of FedCFC-fBind makes it suitable for battery-based on-device FL because wireless transceiver is power-intensive. Note that the communication overhead of FedCFC-FedRep is comparable to that of FedCFC-fBind, but FedCFC-fBind achieves higher accuracy and faster convergence speed.

C. Evaluation on Scalability Regarding Network Size

Due to power constraints and the nature of collected data, MCU-based devices typically operate with a low data-collection rate over time. As a result, many devices need to participate in FL to aggregate sufficient training data. In this section, we evaluate the impact of the number of participating FL clients on the performance of FedCFC-fBind and other FL approaches. We use two metrics: the final model accuracy and the time used on the server in each federation round. We use the NTU RGB+D dataset for evaluation. We control the training data per round to a fixed batch size of 32 samples and limit training to a single epoch. The number of clients increases from 10 to 90.

Fig. 9a shows that FedCFC-fBind consistently achieves the highest accuracy. Although all algorithms experience a decline in accuracy when the number of clients exceeds 40 due to increased model divergence caused by increased data heterogeneity, FedCFC-fBind exhibits only a 2.8% decrease in accuracy, demonstrating its better scalability regarding network size. Fig. 9b shows that as the number of clients increases, the computational time of FedCFC-fBind remains comparable to that of simpler baseline algorithms (i.e., FedProx). This is primarily because FedCFC-fBind avoids transmitting the full model weights, thereby reducing the computational overhead associated with matrix operations. The worst-case computational complexity for solving semi-definite programming problems described in §IV-D using Splitting Conic Solver (SCS) [41] is $\mathcal{O}((|\mathcal{C}|+1)^M)$, where $|\mathcal{C}|$ denotes the number of clients, and M is a constant dependent on the internal workings of the SCS solver. As a result, the computational overhead of the FedCFC-fBind on the server side scales polynomially with the number of clients $|\mathcal{C}|$.

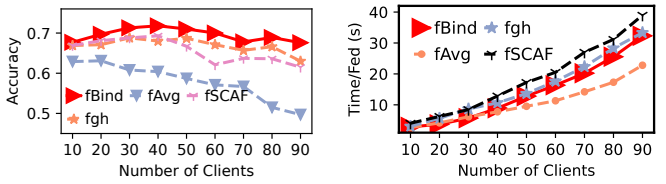
VI. On-Device Experiments

A. Implementations and Deployments

Due to varying resource constraints across the deployed platforms, we adapt the architectures of CFC's three FNN modules accordingly to ensure the model fits within the available memory while maintaining a consistent hidden state dimension $\mathbf{x}$. The CFC model is pre-trained on a subset of the dataset before INT8 quantization. The pre-training stage involves the entire CFC model, including the f, g, and h modules as well as the backbone. To better reflect realistic deployment scenarios, we adopt an incremental learning setting in which only a subset of labels from the full dataset is initially available for training. As new label classes arrive, the system continues training the model through on-device federated learning, thereby simulating the continual adaptation process commonly encountered in IoT applications. We

TABLE III: MCU specification and on-device training resource usage profile of FedCFC-fBind.

Platform	MCU rate (MHz)	SRAM (KB)	f,g,h #layer	CFC size (KB)	Initial acc	End acc	Training time (ms)	Avg SRAM usage (KB)
Nano Sense	64	256	2, 2, 2	6.7	0.73	0.86	347.55	159.76
Nano ESP32	240	320	2, 2, 2	6.8	0.72	0.85	254.47	159.42
M4 Express	120	192	2, 1, 1	4.6	0.72	0.84	300.88	127.18
STM32F303	72	80	1, 1, 1	3.7	0.71	0.79	249.82	81.67



(a) Impact of the number of clients on the accuracy.

(b) Impact of clients number on federation time per round.

Fig. 9: Impact of the number of clients.

implement the CFC model and the weight update algorithm in C programming language for the client. Table III summarizes the MCU specifications. Specifically, we consider these four MCUs for two reasons. First, they capture the device heterogeneity that naturally arises in FL scenarios. Differences in architecture, toolchains, and supported libraries reflect the diversity that can cause training challenges in FL. For example, as the optimized CMSIS-NN library is unavailable for Nano ESP32, we fall back to portable C programming language kernels to implement the necessary operations. Second, they cover a wide spectrum of hardware capabilities, enabling us to evaluate model efficiency and robustness under realistic memory and computation constraints. The details of memory allocation on the MCUs can be found in our conference paper [26].

B. Evaluation of FedCFC-fBind on MCUs

To evaluate the performance of FedCFC-fBind on MCU-based platforms, we train CFC under FL paradigm using a network of 10 client nodes, including 7 Nano Sense devices and one device from each of the three remaining platforms, with a laptop acting as the parameter server. Clients communicate with the server through BLE (Nano Sense), Wi-Fi (Nano ESP32), and UART (Adafruit M4 Express and STM32F303). The UCI-HAR dataset is used for training, with 40% reserved for pre-training the CFC model. We assess the test accuracy of the pre-trained and quantized CFC, the final test accuracy of clients after FL, training time per communication round, and SRAM usage. Each input sample consists of 125 sensor readings over a 2.54-second interval. Table III presents the results, showing that FedCFC-fBind maintains test accuracy between 79% and 87% across the four platforms. The accuracy is influenced by model size and available SRAM, highlighting the trade-offs between computational efficiency and resource constraints in MCU-based deployments. For reference, the highest reported accuracy on the UCI-HAR dataset in the literature is 90.6% [42]. The observed accuracy gap of 3.6% to 11.6% is primarily due to the necessary compromises imposed by quantization and the use of smaller models to accommodate limited SRAM. Our profiling experiments demonstrate that FedCFC-fBind is adaptable and scalable for deployment on MCU-based platforms, even with 256 kB SRAM or lower, making it a viable solution for resource-constrained systems.

We also measure the STM32F303 MCU's energy consumption profile. In its idle state, it draws a current of 14.3 mA. During the training phase, the MCU reaches a maximum current of 25.7 mA. In the testing phase, the average current is about 22.4 mA. Operating on a 100 mAh battery, the projected lifetime for back-to-back inference activities on the STM32F303 MCU is about 268 minutes.

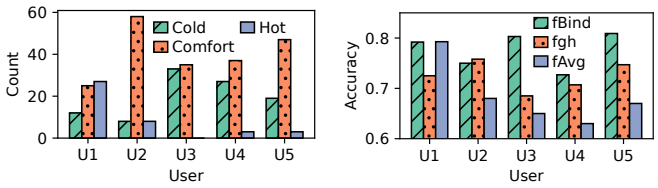
C. Case Study: Thermal Comfort Prediction on MCUs

In this section, we deploy a network of five Arduino Nano Sense at 5 seats in an indoor office environment to construct a personalized thermal comfort prediction model using built-in temperature and light sensors on the Arduino. Thermal comfort perception varies across individuals due to factors such as gender, age, and clothing preferences. As a result, even under the same indoor conditions, different individuals may have distinct and often biased comfort preferences. Therefore, the label distributions across users are highly imbalanced.

Data collection: Each Arduino device records the temperature every hour when a person is present at the corresponding seat. During each sampling period, data is collected for five minutes, and the occupant is asked to report their thermal comfort level. Over a period of 12 days, we collected a total of 342 samples. The distribution of thermal comfort responses for each individual is shown in Fig. 10a. Note that the collected data across different devices shows a noticeable label distribution skew.

Training and inference: We first pre-train the CFC model using 367 samples from the ASHRAE Thermal Comfort Database [43], specifically selecting data from buildings of the same type in our city. The selected features include indoor temperature, outdoor temperature, outdoor humidity, gender, and age. In addition, time is discretized into one-minute intervals and used as an input feature. Subsequently, we use 231 collected samples for on-device federated learning, allowing the model to be further refined based on real-world data from the deployed devices.

Results: Fig. 10b shows that after on-device training, FedCFC-fBind achieves an average accuracy of 77.8%, outperforming FedCFC-fgh, which achieves an accuracy of 70.2%. This result validates the effectiveness of our personalized FL approach and demonstrates that our system can successfully enable on-device FL in real-world scenarios.



(a) Collected thermal comfort distribution of each user.

(b) Thermal comfort prediction of each user.

Fig. 10: Thermal comfort prediction with FedCFC.

VII. Conclusion

This paper highlights the advantages of CFC neural networks for processing time series data on resource-constrained IoT devices. We introduce FedCFC, a CFC-based on-device federated learning network. The design of FedCFC leverages a property identified in this study—namely, the insensitivity of its f sub-model with respect to variations in the training data's class distribution. To address the challenge of local class skews among clients, we propose a novel geometric aggregation strategy, fBind. Extensive evaluations and on-device experiments demonstrate the superior performance of FedCFC-fBind, as well as its adaptability for deployment on low-end devices with as little as 256 kB of memory.

Acknowledgment

This research is supported by the Ministry of Education, Singapore, under its Academic Research Fund Tier 1 RG88/22 and RT14/22.

References

[1] J. V. Jeyakumar, L. Lai, N. Suda, and M. Srivastava, "Sensehar: A robust virtual activity sensor for smartphones and wearables," *SenSys*, 2019.

[2] B. Rim, N.-J. Sung, S. Min, and M. Hong, "Deep learning in physiological signal data: A survey," *Sensors*, vol. 20, no. 4, p. 969, 2020.

[3] X. Wang and E. K. Blum, "Discrete-time versus continuous-time models of neural networks," *J. Comput. Syst. Sci.*, vol. 45, no. 1, pp. 1–19, 1992.

[4] R. Hasani, M. Lechner, A. Amini, D. Rus, and R. Grosu, "Liquid time-constant networks," *AAAI*, 2021.

[5] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," *NeurIPS*, 2018.

[6] R. Hasani, M. Lechner, A. Amini, L. Liebenwein, A. Ray, M. Tschaikowski, G. Teschl, and D. Rus, "Closed-form continuous-time neural networks," *Nature Machine Intelligence*, vol. 4, no. 11, 2022.

[7] X. Qiu, T. Parcollet, J. Fernandez-Marques, P. P. de Gusmao, Y. Gao, D. J. Beutel, T. Topal, A. Mathur, and N. D. Lane, "A first look into the carbon footprint of federated learning." *J. Mach. Learn. Res.*, vol. 24, pp. 129–1, 2023.

[8] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, 2017, pp. 1273–1282.

[9] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *MLSys*, 2020.

[10] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," *ICML*, 2020.

[11] A. Li, J. Sun, B. Wang, L. Duan, S. Li, Y. Chen, and H. Li, "Lotteryfl: Empower edge intelligence with personalized and communication-efficient federated learning," *Symp. Edge Comput.*, 2021.

[12] X. Shuai, Y. Shen, S. Jiang, Z. Zhao, Z. Yan, and G. Xing, "Balancefl: Addressing class imbalance in long-tail federated learning," *IPSN*, 2022.

[13] Y. Zhang, T. Gu, and X. Zhang, "Mdldroid: A chainsgd-reduce approach to mobile deep learning for personal mobile sensing," *IEEE/ACM Transactions on Networking*, vol. 30, no. 1, pp. 134–147, 2021.

[14] S. Saha, S. Sandha, and M. Srivastava, "Machine learning for microcontroller-class hardware: A review." *IEEE Sensors Journal*, vol. 22, no. 22, 2022.

[15] J. Lin, L. Zhu, W.-M. Chen, W.-C. Wang, C. Gan, and S. Han, "On-device training under 256kb memory," in *NeurIPS*, 2022.

[16] Y. Jiang, S. Wang, V. Valls, B. J. Ko, W.-H. Lee, K. K. Leung, and L. Tassiulas, "Model pruning enables efficient federated learning on edge devices," *IEEE Trans. Neural Netw. Learning Syst.*, 2022.

[17] A. Reisizadeh, I. Tziotis, H. Hassani, A. Mokhtari, and R. Pedarsani, "Straggler-resilient federated learning: Leveraging the interplay between statistical accuracy and system heterogeneity," *IEEE J. Sel. Areas Inf. Theory*, vol. 3, no. 2, pp. 197–205, 2022.

[18] M. Duan, D. Liu, X. Chen, Y. Tan, J. Ren, L. Qiao, and L. Liang, "Astraea: Self-balancing federated learning for improving classification accuracy of mobile deep learning applications," *ICCD*, 2019.

[19] L. Wang, S. Xu, X. Wang, and Q. Zhu, "Addressing class imbalance in federated learning," *AAAI*, 2021.

[20] L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai, "Exploiting shared representations for personalized federated learning," *PMLR*, 2021.

[21] X. Ouyang, Z. Xie, J. Zhou, J. Huang, and G. Xing, "Clusterfl: A similarity-aware federated learning system for human activity recognition," *MobiCom*, 2021.

[22] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *IEEE Trans. Neural Netw. Learning Syst.*, 2022.

[23] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning: A meta-learning approach," *arXiv*, 2020.

[24] H. Xu, P. Zhou, R. Tan, and M. Li, "Practically adopting human activity recognition," *MobiCom*, 2023.

[25] C. Koch and I. Segev, *Methods in neuronal modeling*. MIT press, 1998.

[26] Y. Dai and R. Tan, "Fedcfc: On-device personalized federated learning with closed-form continuous-time neural networks," in *IPSN*, 2024.

[27] Attila, Indlekofera, and P. Schmidt, "PPG-DaLiA," UCI Machine Learning Repository, 2019, DOI: https://doi.org/10.24432/C53890.

[28] "Human Activity Recognition Using Smartphones," 2012. [Online]. Available: https://doi.org/10.24432/C54S4K

[29] D. W. Romero, A. Kuzina, E. J. Bekkers, J. M. Tomczak, and M. Hoogendoorn, "Ckconv: Continuous kernel convolution for sequential data," *ICLR*, 2021.

[30] A. Johnson, T. Pollard, L. Shen, L.-w. Lehman, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. Celi, and R. Mark, "Mimic-iii, a freely accessible critical care database," *Scientific Data*, vol. 3, 05 2016.

[31] M.-A. Tnani, M. Feil, and K. Diepold, "Smart data collection system for brownfield cnc milling machines: A new benchmark dataset for data-driven machine monitoring," *Procedia CIRP*, vol. 107, 2022.

[32] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," *ACM MM*, 2014.

[33] H. H. Sohrab, *Basic real analysis*. Springer, 2003, vol. 231.

[34] [Online]. Available: https://courses.cs.washington.edu/courses/cse599i/20au/resources/L12_duality.pdf

[35] N. Aronszajn, "Theory of reproducing kernels," *Transactions of the American mathematical society*, vol. 68, no. 3, pp. 337–404, 1950.

[36] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural computation*, vol. 10, no. 5, pp. 1299–1319, 1998.

[37] H. Ichino, K. Kaji, K. Sakurada, K. Hiroi, and N. Kawaguchi, "Hascpac2016: large scale human pedestrian activity corpus and its baseline recognition," *UbiComp*, 2016.

[38] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang, "Ntu rgb+ d: A large scale dataset for 3d human activity analysis," *CVPR*, 2016.

[39] J. A. Miranda-Correa, M. K. Abadi, N. Sebe, and I. Patras, "Amigos: A dataset for affect, personality and mood research on individuals and groups," *IEEE Trans. Affective Comput.*, vol. 12, no. 2, 2018.

[40] L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai, "Exploiting shared representations for personalized federated learning," in *ICML*, 2021.

[41] B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd, "Conic optimization via operator splitting and homogeneous self-dual embedding," *Journal of Optimization Theory and Applications*, 2016.

[42] E. Eldele, M. Ragab, Z. Chen, M. Wu, C. K. Kwoh, X. Li, and C. Guan, "Time-series representation learning via temporal and contextual contrasting," *arXiv*, 2021.

[43] V. F. Ličina, T. Cheung, H. Zhang, R. De Dear, T. Parkinson, E. Arens, C. Chun, S. Schiavon, M. Luo *et al.*, "Development of the ashrae global thermal comfort database ii," *Building and Environment*, 2018.

Yimin Dai is a Ph.D candidate at the College of Computing and Data Science, Nanyang Technological University (NTU), Singapore. He is also with the Interdisciplinary Graduate School of NTU. He received the M.Eng. (2023) from NTU and received the B.Sc. (2021) from The Hong Kong Polytechnic University, Hong Kong, China. His research interests include designing resource-efficient and stable continuous-time neural networks for edge AI systems. He is a Student Member of IEEE.

Rui Tan is a full professor at College of Computing and Data Science, Nanyang Technological University, Singapore. Previously, he was a Research Scientist (2012-2015) and a Senior Research Scientist (2015) at Advanced Digital Sciences Center of University of Illinois at Urbana-Champaign, and a postdoctoral Research Associate (2010-2012) at Michigan State University. He received the Ph.D. (2010) degree in computer science from City University of Hong Kong, the B.S. (2004) and M.S. (2007) degrees from Shanghai Jiao Tong University. His research interests include cyber-physical systems and Internet of things. He is the recipient of Best Demo Award from SenSys'24, Best Paper Awards from ICCPS'23 and IPSN'17. He served as Associate Editor of IEEE Transactions on Mobile Computing and ACM Transactions on Sensor Networks, TPC Co-Chair of e-Energy'23, EWSN'24, SenSys'24, and General Co-Chair of e-Energy'24 and RTCSA'25. He received the Distinguished TPC Member recognition from SenSys in 2025 and from INFOCOM in 2017, 2020, and 2022. He is a Senior Member of IEEE.