H-FedSN: Personalized Sparse Networks for Efficient and Accurate Hierarchical Federated Learning for IoT Applications

Jiechao Gao[1,*], Yuangang Li[2,*], Jie Wang[1],
Michael Lepech[1], Yue Zhao[3], Brad Campbell[4]
[1]Stanford University, Stanford, CA, USA
[2]University of California, Irvine, Irvine, CA, USA
[3]University of Southern California, Los Angeles, CA, USA
[4]University of Virginia, Charlottesville, VA, USA
[*]Corresponding author: jiechao@stanford.edu, yuanganl@uci.edu

Abstract

With the rapid development of the Internet of Things (IoT), federated learning (FL) has gained increasing attention for its privacy-preserving use of distributed data. However, conventional two-tier FL architectures are poorly suited to the hierarchical and heterogeneous nature of real-world IoT systems. Hierarchical Federated Learning (HFL) introduces multi-layer aggregation to better match IoT environments, but still suffers from communication inefficiencies and performance limitations caused by large data transfers, non-IID data distributions, and uneven device participation. These challenges hinder the realization of low-latency and high-accuracy training in practical IoT deployments. To address these limitations, we propose H-FedSN for practical IoT environments. H-FedSN leverages a binary mask mechanism with shared and personalized layers to reduce communication overhead by creating a sparse network without altering original weights. To tackle data heterogeneity and imbalanced device distribution, H-FedSN incorporates personalized layers for local data adaptation and employs Bayesian aggregation with cumulative Beta distribution updates at edge and cloud levels, effectively balancing contributions from diverse client groups. Experiments on three real-world IoT datasets and MNIST under non-IID conditions show that H-FedSN reduces communication costs by up to 477 times compared to baseline methods while maintaining high accuracy, making it well-suited for hierarchical FL in IoT deployments.

Introduction

Federated learning (FL) enables multiple distributed devices to collaboratively train a model without sharing raw data, allowing global knowledge integration while preserving data privacy [1, 2]. Integrating FL into ubiquitous Internet of Things (IoT) environments unlocks a wide range of applications [3, 4]. As IoT systems continue to expand in scale and complexity, the data collection, computing, and communication resources follows a natural multi-tier structure[5, 6]. Terminal devices operate at the sensing layer and continually generate data; edge servers provide intermediate computation and aggregation; and cloud servers execute global coordination and optimization. For example, in smart city surveillance, cameras connect to building-level servers before interacting with the cloud [7]; in smart agriculture, drones and ground sensors upload data to farm-level gateways that then communicate with regional servers [8]; and in healthcare monitoring, wearable devices transmit physiological signals to nearby edge gateways before being forwarded to the cloud [9]. Given this intrinsic layering of IoT systems, collaborative learning frameworks generally follow the corresponding collaboration pathways among terminals, edges, and the cloud.

Compared to two-tier federated learning, hierarchical federated learning (HFL) [10] is inherently more suitable for multi-tier architectures of IoT systems and is thus more practical for real-world deployments. However, it still faces key challenges in communication efficiency and model accuracy. In particular, bandwidth constraints and communication latency pose critical bottlenecks that degrade HFL's training efficiency and convergence speed across tiers. Limited bandwidth restricts the transmission of model parameters, prolonging training rounds, while high latency introduces cascading delays during hierarchical aggregation. For example, HierFAVG [10] requires devices to transmit full model parameters over bandwidth-limited IoT links across multiple layers, making real-world deployment impractical. Additionally, IoT devices typically encounter non-independent and identically distributed (non-IID) data and imbalanced client distributions, which further degrade model accuracy [11]. Therefore, designing communication-efficient HFL strategies that mitigate latency and bandwidth limitations while maintaining high model accuracy is essential to enable scalable and high-performance IoT applications [12].

Although many approaches [13, 14, 15] have been proposed to enhance communication efficiency in conventional two-tier federated learning, adapting these methods to IoT scenarios, particularly in three-tier hierarchical federated learning (HFL), remains challenging due to limited research on architectural adapta-

tion and the risk of performance degradation. For example, quantization-based techniques [15] may accumulate errors across multiple aggregation layers, significantly impairing model accuracy. In contrast, personalized federated learning algorithms [16, 17, 18] are well-suited to address data heterogeneity and can be integrated into HFL frameworks with relative ease, offering improved accuracy. However, these methods typically neglect communication efficiency. HFL-based methods also face limitations in balancing accuracy and communication efficiency, such as HierFAVG [10] that, despite fitting hierarchical IoT architectures, transmits full model parameters incurring high communication costs, and ESPerHFL [19] that, although achieving better accuracy via personalization, introduces additional parameters thus increasing communication overhead.

To balance communication costs and accuracy, we propose H-FedSN, designed for real-world IoT environments to achieve low communication costs and high accuracy in the HFL architecture. H-FedSN centers on a personalized, structured sparse model for each client. Clients start with an identical neural network with frozen weights and train a corresponding binary (0,1) mask network. This binary mask network has the same structure as the original neural network. Applying this binary mask to the original network determines which connections to retain or prune, thereby creating a sparse network. The mask network consists of shared layers and personalized layers. Shared layers participate in global aggregation, becoming identical across all clients after training. Personalized layers are trained on client-specific data, remain local, and do not participate in global aggregation, resulting in unique layers for each client. To further enhance model accuracy and address imbalanced client distribution, H-FedSN incorporates Bayesian aggregation at both edge and cloud levels. This method uses cumulative updates of Beta distribution parameters to balance contributions from edge nodes with varying numbers of connected clients. While edge nodes with more clients have a larger immediate impact on parameter updates, the cumulative nature of these updates ensures a fair representation of all edge nodes over time. Combined with the locally retained personalized mask layers, this approach allows the system to adapt to imbalanced client distributions across edges and achieve high model accuracy.

Experiments on three real-world IoT datasets and MNIST under non-IID conditions, we compare our H-FedSN with 6 baselines, show that the communication cost of H-FedSN was lower than all baseline methods, reducing it by up to 477 times compared to baseline methods. Regarding accuracy, H-FedSN achieves much better results by an average of 8.6% compared to communication-enhanced FL methods and is comparable to personalized FL methods within HFL architecture.

Results

We empirically evaluate the performance of H-FedSN, focusing on both accuracy and communication costs. Our experimental analysis covers four distinct datasets: the widely-utilized MNIST [20], supplemented by three IoT-specific datasets: WISDM (watch) [21], WIDAR [22, 23], and WISDM (phone) [24]. MNIST acts as a general benchmark, while the inclusion of IoT datasets showcases our method's applicability in real-world IoT settings.

We compare H-FedSN against 6 baselines adapted for the HFL architecture. ESPerHFL [19], HierFAVG [10] naturally supports HFL, whereas FedPer [18], FedRS [16], FedCAMS [25], and TOPK [26], initially designed for standard FL, have been modified to fit an HFL architecture.

Datasets Setting

Table 1 shows the statistical distribution of the datasets used in our experiments. Below, we describe each dataset along with its preprocessing methods:

Table 1: Statistical information of datasets.

Dataset	**Dimension**	**# Samples (Train / Test)**	**Classes**	#Fixed Classes per Client
MNIST	28 x 28	500,000 / 10,000	10	6
WISDM (WATCH)	200 x 6	16,569 / 4,103	12	7
WIDAR	22 x 20 x 20	11,372 / 5,222	9	5
WISDM (PHONE)	200 x 6	13,714 / 4,073	12	7

- MNIST comprises handwritten digits across 10 classes, widely used for image classification model training [20]. Preprocessing involved converting images to tensors and normalizing pixel values to [-1, 1].

- WISDM (WATCH) [21] encompasses accelerometer and gyroscope readings from smartwatches, capturing 18 daily activities performed by 51 participants. Our preprocessing approach aligned with established methods in the field of sensor-based activity recognition [27, 28, 29]. We segmented each 3-minute recording using a sliding window technique, employing a 10-second window with 50% overlap. Subsequently, we applied normalization to each dimension of the resulting samples, centering the data around zero and adjusting the scale to achieve unit variance.

- WIDAR [22, 23], designed for contactless gesture recognition using Wi-Fi signal strength, includes data from 17 participants performing 22 unique gestures. We

applied body velocity profiling (BVP) to account for environmental variations, followed by scalar normalization. The final representation was structured into $22 \times 20 \times 20$ samples, covering the time axis and x-y velocity features. Only gestures recorded by more than three users were retained to ensure consistency between training and testing sets.

- WISDM (PHONE) [24] consists of accelerometer data obtained from smartphones during participants' daily activities. We implemented an identical preprocessing protocol to that used for the WISDM (WATCH) dataset. This involved segmenting the data with a 10-second sliding window (50% overlap), followed by normalization of each dimension to zero mean and unit variance.

We simulate non-IID data distributions using a label imbalance approach that ensures each client has a fixed set of labels, following FedAvg [2] and further refined by Li et al.[30]. Specifically, each client is randomly assigned n distinct label IDs, and the samples of each label are distributed equally among the clients possessing that label. This guarantees every client consistently holds data from exactly n labels with no overlap among clients. We apply the same label partitioning to both training and test sets to maintain consistent levels of label imbalance. Table 1 shows the total number of classes for each dataset and the fixed number of classes per client. For illustration, Fig. 5 depicts how WIDAR is partitioned among 5 clients.

Experimental Design

We used a 4-layer convolutional neural network (CNN) comprising two convolutional blocks (each with two convolutional layers followed by a pooling layer, with 64 filters per layer in the first block and 128 in the second) and three fully-connected layers (two with 256 units each, followed by an output layer matching the dataset's number of classes). For H-FedSN, we set the last three layers as private layers, which do not participate in the aggregation process.

To comprehensively evaluate the performance of H-FedSN, we compare H-FedSN against 6 baselines:

- ESPerHFL [19]: Implements edge-server personalization in HFL by using learnable mixing parameters and similarity-based aggregation with Tanimoto coefficients.
- HierFAVG [10] Hierarchical federated averaging with client-edge-cloud structure.

- FedPer [18] Federated personalization keeping part of model parameters local.
- FedRS [16] Addresses label distribution skew using Restricted Softmax.
- FedCAMS [25] Combines gradient compression with adaptive optimization to reduce communication overhead.
- TOPK [26] Sparsification method selecting largest k (3.125%) elements of gradients.

All baselines are implemented with aggregations at both edge and cloud levels in the HFL architecture.

In addition, we adopted the same base model configuration for each baseline method, which is the same 4-layer CNN (CONV-4) used for H-FedSN. We also applied the same data configurations, ensuring consistency across all experiments with the non-IID data distribution.

To comprehensively assess the efficacy of our proposed algorithm, we evaluate the following key performance metrics:

- Inference Performance: We measure inference performance in two ways: (1) **Average Accuracy**: Defined as the mean accuracy across all clients, this metric quantifies the overall inference performance of the federated model. It serves as the primary benchmark for comparing methods, clearly indicating the algorithm's effectiveness in achieving accurate predictions on distributed, heterogeneous datasets. (2) **Accuracy Distribution across Clients**: In addition to the average accuracy, we visualize the distribution of inference accuracies among individual clients using boxplots. This metric captures the consistency and robustness of the algorithm across heterogeneous and imbalanced client distributions, reflecting its practical reliability in realistic IoT environments.
- Communication Cost: We measure the volume of data (in KB) transmitted per training round between clients and edge servers, and from edge servers to the cloud. Minimizing communication overhead is critical in real-world IoT scenarios due to bandwidth limitations, directly affecting system efficiency, scalability, and latency.

To illustrate the adaptability of our proposed algorithm across different scenarios and environments, we designed the following experimental configurations:

- E2C5: 2 edge nodes and 5 client nodes. This minimalistic setup tests the algorithm's effectiveness in small-scale deployments, such as some small offices or local retail stores with limited devices and minimal infrastructure [31].
- E20C50: 20 edge nodes and 50 client nodes. This configuration evaluates scalability and robustness in larger, complex network topologies, simulating environments like a smart city surveillance system across multiple buildings [7].
- E2C50: 2 edge nodes and 50 client nodes. This setting simulates high client density scenarios, such as a smart healthcare system for remote patient monitoring where numerous wearable devices and home health monitors connect to local healthcare facility servers [32].
- Imbalanced E5C50: 5 edge nodes with 50 clients distributed in ratios of 0.4, 0.2, 0.2, 0.1, and 0.1. This imbalanced setup tests consistency and fairness across unevenly distributed network nodes, representing scenarios like smart agriculture systems with varying device densities across different farm sections [8].

For all experiments, the hyperparameters were consistently set as follows: Global Rounds = 200, the learning rates for H-FedSN were 0.01 for all datasets except the WIDAR dataset, where the learning rate was 0.035. For the baseline algorithms, the learning rate was uniformly set to 0.001 (see Supplementary Table 1).

Analysis

In this section, we present and analyze the results obtained from our experiments. We provide a comparative evaluation of H-FedSN against the baselines in terms of inference performance and communication costs.

In the **E2C5** experimental setting, we conduct a comprehensive comparison across three categories of methods: personalization-enhanced approaches (FedPer, FedRS), communication optimization approaches (FedCAMS, TOPK), and the standard hierarchical federated averaging baseline (HierFAVG). As shown in Table. 2, we evaluate each method in terms of the average accuracy (avg Acc.) across all clients (higher is better) and the per-round communication cost (Comm.) measured in KB (lower is better). Among all baselines, the personalization-enhanced methods typically achieve high accuracies, so we compare H-FedSN against each dataset's strongest personalization baseline to gauge how closely it can match their accuracy at significantly lower overhead. For WISDM(WATCH) dataset, FedRS

Table 2: Performance comparison of H-FedSN and baseline methods across four datasets in E2C5 configuration. Methods are evaluated under two metrics, with avg Acc. (average accuracy of clients; higher (↑), the better) and Comm. (communication cost; lower (↓), the better) as the metrics. The **Best** results are highlighted in bold, the second-best results are double-underlined, and the third-best results are single-underlined.

Algorithms	MNIST		WISDM(WATCH)		WIDAR		WISDM(PHONE)	
	avg Acc. ↑	Comm. (KB) ↓	avg Acc. ↑	Comm. (KB) ↓	avg Acc. ↑	Comm. (KB) ↓	avg Acc. ↑	Comm. (KB) ↓
ESPerHFL	0.9963	120828.63	0.7005	14168.75	0.7532	72416.56	0.3882	14168.75
HierFAVG	0.9968	60414.31	0.7382	7084.38	0.7789	36208.28	0.3707	7084.38
Personalization-enhanced FL methods								
FedPer	0.9950	60334.00	0.7246	6988.00	**0.8285**	36136.00	**0.4836**	6988.00
FedRS	0.9965	60414.31	**0.7597**	7084.38	0.7432	36208.28	0.3814	7084.38
Communication-optimizated FL methods								
FedCAMS	0.9929	1887.98	0.5965	211.42	0.7097	1131.54	0.3967	211.42
TOPK	0.9908	1887.98	0.5977	211.41	0.6218	1131.53	0.3002	211.41
H-FedSN(**Our**)	0.9936	**252.94**	0.7442	**121.88**	0.7637	**264.75**	0.4298	**121.88**

attains an accuracy of 0.7597 (the highest among baselines), whereas H-FedSN reaches 0.7442, only slightly lower, yet FedRS's communication cost (7084.38 KB) is 58 times larger than H-FedSN's 121.88 KB. A simliar pattern also appears in WIDAR and WISDM(PHONE) dataset. In WIDAR, FedPer's accuracy surpassing H-FedSN a little, but it transmits 36136 KB compared to H-FedSN's 264.75 KB, over a 136-fold difference. In WISDM(PHONE), where FedPer exceeds H-FedSN a little, yet requires more than 57 times the communication cost (6988 KB vs. 121.88 KB). Finally, although all baselines achieve relatively similar average accuracy on **MNIST**, H-FedSN remains below 300KB in per-round communication. In all these comparisons, H-FedSN delivers near-top accuracy while reducing communication by one to two orders of magnitude, thanks to transmitting compact binary masks for shared layers, retaining personalized parameters entirely local, and employing Bayesian aggregation for robust updates.

In Figure 7, we illustrate the training of six methods (HierFAVG, FedPer, FedRS, FedCAMS, TOPK, and H-FedSN) in HFL architecture. From the plotted curves, it is evident that all methods reach near-convergence around the 20th round, with only minor accuracy fluctuations thereafter, indicating that their global models stabilize at this stage. Notably, TOPK and FedCAMS exhibit non-monotonic behavior in their accuracy curves. TOPK's accuracy rises rapidly in the early rounds but then undergoes a marked drop before gradually recovering. This pattern emerges because TOPK transmits only the largest gradient components,

initially boosting accuracy but overlooking smaller yet important updates. As these neglected updates are later transmitted through error compensation, the model corrects its early bias, causing a dip and eventual rebound. By contrast, FedCAMS combines adaptive optimization with gradient compression, resulting in more moderate updates during the initial phase. Nevertheless, as training proceeds, conflicting gradients arising from multi-layer data heterogeneity lead to a mid-round accuracy dip, which is then partially offset by FedCAMS's adaptive feedback mechanism. Despite these transient fluctuations, both methods ultimately achieve convergence on par with the other approaches in our experiments.

We next examine more demanding network conditions with larger scales, uneven client distributions, and more complex connectivity. The subsequent experiments (E2C50, Imbalanced E5C50, E20C50) compare H-FedSN against four representative baselines spanning two categories: FedPer and FedRS for personalization, and FedCAMS and TOPK for communication efficiency. As shown in the following sections, H-FedSN not only delivers lower per-round transmission cost than FedCAMS and TOPK along with higher average accuracy and more consistent per-client performance, but also matches the accuracy of FedPer and FedRS at roughly two orders of magnitude lower transmission cost.

In the **Imbalanced E5C50** configuration, where client distribution is uneven, H-FedSN demonstrates its robustness by maintaining high performance across varying client densities while significantly reducing communication costs across all datasets.

As shown in Fig. 6 and Fig. 8, on the wisdm datasets, H-FedSN matches the strongest personalization baseline FedPer (WISDM(WATCH): 71.13% versus 71.03%; WISDM(PHONE): 46.86% versus 47.25%). On WIDAR, FedPer (78.29%) and FedRS (77.19%) lead, with H-FedSN at 72.34%. Across all three datasets, H-FedSN transmits roughly two orders of magnitude less per round than the personalization baselines and several times less than FedCAMS and TOPK, and surpasses both communication-optimized baselines on accuracy and per-client consistency. The personalized layers retained locally and the Bayesian aggregation propagating only sparse binary masks across the hierarchy together absorb the uneven client distribution.

In the **E20C50** configuration, H-FedSN maintains minimal communication costs while achieving strong inference accuracy across clients.

As shown in Fig.9 and Fig.6, H-FedSN stays close to these personalization baselines (FedPer, FedRS) but transmits roughly two orders of magnitude less per round. The communication-optimized baselines (FedCAMS, TOPK) trail H-FedSN on every dataset both in mean accuracy and per-client consistency. H-

FedSN occupies the low-communication corner of the accuracy–communication Pareto frontier, achieving the lowest transmission cost while staying within a small accuracy margin of the highest-accuracy methods.

The **E2C50** configuration, where each edge server coordinates a high density of clients, places a considerable strain on network resources. In Fig.10, FedPer reaches the highest mean accuracy on every dataset, and H-FedSN stays close to these personalization baselines. FedCAMS and TOPK fall below H-FedSN on every dataset. Per-round transmission cost of H-FedSN (Fig. 6) stays roughly two orders of magnitude below FedPer and FedRS and several times below FedCAMS and TOPK.

Table 3: Accuracy on MNIST Across Experiment Configurations (E2C50, imbalanced E5C50, E20C50)

Setting	**FedCAMS**	**TOPK**	**H-FedSN**
Imbalanced E5C50	0.9956	0.9936	0.9893
E2C50	0.9964	0.9941	0.9878
E20C50	0.9967	0.9960	0.9908

Although we also evaluate the methods on **MNIST** in the subsequent E2C50, imbalanced E5C50, and E20C50 configurations, their accuracies differ by less than 1% across all baselines, offering little additional insight beyond what is already established under E2C5. As summarized in Table 3, all methods reach near-identical accuracy on MNIST, while H-FedSN maintains the lowest communication overhead. Hence, we omit dedicated MNIST accuracy curves in the following figures to avoid redundancy, but we include this table to confirm that the same efficiency advantage of H-FedSN holds in those larger-scale or imbalanced settings.

In summary, H-FedSN effectively balances communication efficiency and inference accuracy, consistently outperforming baseline methods across four IoT scenarios. Its personalized sparse structure and Bayesian aggregation strategy significantly reduce communication overhead while maintaining robust performance under data heterogeneity and imbalanced client distributions. These results highlight the practical advantages of H-FedSN for IoT-oriented hierarchical federated learning.

Discussion

This work set out to address the dual challenges of communication efficiency and inference accuracy in hierarchical federated learning for practical IoT environments. Our experimental evaluations show that H-FedSN achieves inference accuracy comparable to or exceeding state-of-the-art personalization-enhanced baselines, while reducing per-round communication costs by one to two orders of magnitude. These gains hold consistently across small-scale, large-scale, high-density, and imbalanced network configurations, confirming the robustness of the proposed approach. Below, we discuss the implications of these findings and situate them within the broader literature.

- ***H-FedSN addresses fundamental limitations of conventional FL in IoT scenarios.*** Federated learning has transformed distributed machine learning by enabling collaborative model training without centralizing raw data, with foundational algorithms such as FedAvg [2], FedProx [33], and Scaffold [34] establishing effective cross-node collaboration. However, these traditional FL approaches rely on a central server for aggregation, which introduces critical limitations in real-world IoT deployments: scalability bottlenecks as the number of devices grows into the thousands or millions [35], communication delays exacerbated by geographic dispersion [36], rapid depletion of limited computational and energy resources on IoT devices through frequent long-distance communication [37], and a single point of failure that can disrupt the entire learning process [38]. H-FedSN addresses these issues through its hierarchical architecture, which distributes aggregation across edge and cloud layers, reducing reliance on a single central server and better matching the inherent multi-tier structure of IoT systems.

- ***The hierarchical architecture introduces new challenges that H-FedSN is specifically designed to overcome.*** Hierarchical Federated Learning (HFL) has been proposed to address scalability limitations by introducing multi-layer aggregation across client, edge, and cloud tiers, as exemplified by the HierFAVG algorithm that organizes clients into a hierarchical, tree-like topology naturally suited to geographically distributed IoT environments [10]. While this multi-level structure improves scalability, it also introduces new challenges in communication efficiency and cross-layer coordination [12]. Prior work on communication efficiency in conventional FL has explored strategies such as structure-based communication reduction (e.g., FedSCR, which reduces upstream communication by nearly 50% [39]), communication-computation trade-off balancing [40],

and network pruning [41]. However, these methods were designed for two-tier settings and do not readily transfer to HFL's multi-level structure, where errors such as quantization noise can accumulate across aggregation layers [15] and the unique requirements at each tier demand tailored solutions. This leaves a significant gap in the literature regarding communication optimization specifically within HFL. H-FedSN fills this gap by introducing a dual-layer binary masking mechanism that operates natively within the hierarchical architecture, transmitting only compact binary masks rather than full model weights or gradients at each level.

- ***The combination of sparsification, personalization, and Bayesian aggregation provides complementary benefits.*** Unlike existing approaches that address communication efficiency or data heterogeneity in isolation, H-FedSN integrates three complementary mechanisms. Network sparsification through binary masks reduces communication costs by up to 477 times compared to baseline methods. The separation of shared and personalized mask layers allows the model to adapt to heterogeneous, non-IID local data distributions while retaining collaborative benefits through global aggregation of shared layers, an advantage shared with personalization-focused methods such as FedPer [18] and FedRS [16], but achieved here without transmitting full model parameters. Bayesian aggregation at both edge and cloud levels leverages cumulative Beta distribution updates to robustly balance contributions from edge nodes with varying numbers of connected clients, effectively handling the imbalanced device participation common in practical IoT deployments. Our experimental results across four datasets and four network configurations confirm that this integrated design enables H-FedSN to match the accuracy of the strongest personalization-enhanced baselines while requiring roughly two orders of magnitude less communication per round, and to deliver both lower transmission costs and higher accuracy compared to communication-optimized methods such as FedCAMS [25] and TOPK [26].

- ***Practical implications for IoT deployment.*** The consistent performance of H-FedSN across the E2C5, E20C50, E2C50, and Imbalanced E5C50 configurations demonstrates its applicability to diverse real-world IoT scenarios ranging from small offices to smart city surveillance systems. The dramatic reduction in communication overhead is particularly significant for bandwidth-constrained IoT environments, where transmitting full model parameters across multiple aggregation layers is often impractical. By transmitting only binary masks for shared layers and retaining personalized parameters entirely locally, H-FedSN

minimizes the data volume traversing the network while preserving each client's ability to adapt to its local data characteristics.

Several aspects of the current study point to promising extensions. Our experiments are conducted on a three-tier hierarchy with a single model architecture (CONV-4); scaling H-FedSN to scenarios with more than three aggregation layers and to deeper or heterogeneous model architectures would further test its generality. The current sparsity level is fixed throughout training; designing adaptive mechanisms that dynamically adjust sparsity levels according to real-time network conditions and client capabilities could improve efficiency in highly dynamic IoT environments. Additionally, exploring alternative mask parameterizations beyond the current Bernoulli-based approach and broader families of Bayesian priors may yield further gains in both accuracy and convergence speed.

Methods

Overview

H-FedSN is an algorithm based on the HFL architecture, designed to optimize communication efficiency and ensure model accuracy through efficient mask training, personalized layers, and Bayesian aggregation. Zhou et al. demonstrated that training sparse models with masks, while keeping the model parameters frozen, is an effective alternative to traditional training methods [42].

We consider a three-tier hierarchical federated learning system with $|K|$ clients, $|E|$ edge servers, and a single cloud server. Each edge $e \in E$ serves a disjoint subset of clients $K_e \subseteq K$ (with $K = \bigcup_{e \in E} K_e$). Each client k owns a private, potentially non-IID dataset D_k that never leaves the device. Given a frozen shared initialization $\mathbf{w}_{\text{init}} \in \mathbb{R}^d$, each client k trains a personalized probability mask $\boldsymbol{\theta}_k \in [0,1]^d$. Its stochastic binarization $\mathbf{m}_k \sim \text{Bern}(\boldsymbol{\theta}_k) \in \{0,1\}^d$ decomposes as $\mathbf{m}_k = (\mathbf{m}_k^s, \mathbf{m}_k^p)$ over shared and client-private layers, and induces a sparse model $\dot{w}_k = \mathbf{m}_k \odot \mathbf{w}_{\text{init}}$. The system jointly trains the per-client masks $\{\boldsymbol{\theta}_k\}_{k \in K}$ to minimize each client's expected local task loss on its private data D_k, subject to three communication and privacy constraints: (i) raw data D_k never leaves client k; (ii) each client $k \in K_e$ uploads only the shared binary mask $\mathbf{m}_k^s$ to its edge server e, and each edge e uploads only its aggregated edge mask to the cloud; (iii) the private mask $\mathbf{m}_k^p$ is retained on client k throughout training and never aggregated.

Fig. 1 depicts an overview of the proposed H-FedSN framework. H-FedSN allows each client to freeze its random initialized local model parameters $\mathbf{w}_{\text{init}} =$

$(w_{\text{init}}^1, w_{\text{init}}^2, \ldots, w_{\text{init}}^d) \in \mathbb{R}^d$ while collaboratively training a personalized probability mask $\boldsymbol{\theta}_k \in [0,1]^d$ for each client k, using their respective local datasets D_k. This probability mask $\boldsymbol{\theta}_k$ determines the activation probability of the local model parameters and serves as the Bernoulli parameter for the stochastic binary mask $\mathbf{m}_k \sim \textbf{Bern}(\boldsymbol{\theta}_k) \in \{0,1\}^d$ (①-a). The binary mask $\mathbf{m}_k$ is then applied to the frozen parameters $\mathbf{w}_{\text{init}}$ to create a structured sparse model, uniquely personalized for each client (④). The resulting model, expressed as $\dot{w}_k = \mathbf{m}_k \odot \mathbf{w}_{\text{init}}$, is optimized to enhance performance on specific tasks. For a more detailed description of the algorithm, refer to Algorithm 1.

Algorithm 1 H-FedSN

Require: Number of training rounds T, number of local iterations τ, learning rate η, initial Beta distribution parameters $\lambda_0 = 1$

Ensure: Training a personalized and structured sparse model unique to each client.

At the cloud server, initialize Beta priors $\alpha_g^0 = \beta_g^0 = \lambda_0$, and initialize a random network with weight vector $\mathbf{w}_{\text{init}} \in \mathbb{R}^d$, and then broadcast it to the clients through edges.
At the edge servers, initialize Beta priors $\alpha_e^0 = \beta_e^0 = \lambda_0$.
For all clients, initialize random mask probability $\boldsymbol{s}_{k,1}$
for $t = 1$ to T **do**
  for each edge server $e \in E$ **in parallel do**
    Edge server:
    for each client $k \in K_e$ **in parallel do**
      Client:
      if $t > 1$ **then**
        $\boldsymbol{\theta}_{k,t} \leftarrow \boldsymbol{\theta}_{g,t-1} + \boldsymbol{\theta}_{k,t-1}^p$
        $\mathbf{s}_{k,t} \leftarrow \text{Sigmoid}^{-1}(\boldsymbol{\theta}_{k,t})$
      end if
      for $i = 1$ to τ **do**
        $\boldsymbol{\theta}_{k,t} \leftarrow \text{Sigmoid}(\mathbf{s}_{k,t})$
        $\mathbf{m}_{k,t} \leftarrow \textbf{Bern}(\boldsymbol{\theta}_{k,t})$
        $\dot{w}_{k,t} \leftarrow \mathbf{m}_{k,t} \odot \mathbf{w}_{\text{init}}$
        Compute loss $L(f(\dot{w}_{k,t}), D_k)$
        $\mathbf{s}_{k,t} \leftarrow \mathbf{s}_{k,t} - \eta \nabla L(f(\dot{w}_{k,t}), D_k)$
      end for
      $\boldsymbol{\theta}_{k,t} \leftarrow \text{Sigmoid}(\mathbf{s}_{k,t})$
      $\mathbf{m}_{k,t} \leftarrow \textbf{Bern}(\boldsymbol{\theta}_{k,t})$
      Decompose $\mathbf{m}_{k,t}$ into shared $\mathbf{m}_{k,t}^s$ (over shared layers) and private $\mathbf{m}_{k,t}^p$ (over personalized layers).
      Upload $\mathbf{m}_{k,t}^s$ to edge server e
    end for
    Edge server:
    $\mathbf{m}_{e,t} = \text{BayesianAggregationAtEdge}(\{\mathbf{m}_{k,t}^s\}_{k \in K_e}, t)$
    Upload $\mathbf{m}_{e,t}$ to cloud server
  end for
  Cloud server:
  $\boldsymbol{\theta}_{g,t} = \text{BayesianAggregationAtCloud}(\{\mathbf{m}_{e,t}\}_{e \in E}, t)$
  Broadcast $\boldsymbol{\theta}_{g,t}$ to all edges $e \in E$, and all clients
end for
For all clients, $\mathbf{m}_{k,final} = \textbf{Bern}(\boldsymbol{\theta}_{g,T} + \boldsymbol{\theta}_{k,T}^p)$. Generate the final model: $\dot{w}_{k,final} \leftarrow \mathbf{m}_{k,final} \odot \mathbf{w}_{\text{init}}$.

At the client level, each client k is associated with an edge server e and is part of the set K_e. Here, e represents a specific edge server, and K_e denotes the set of all clients connected to that particular edge server e. This setup implies multiple edge servers, each managing its own group of clients. Clients are responsible for training a probability mask $\boldsymbol{\theta}_k$, which is dimensioned $[0,1]^d$ to match the size of the client's local model parameters. Especially, the probability mask is bifurcated into a shared layer $\boldsymbol{\theta}_k^s \in [0,1]^s$ and a private layer $\boldsymbol{\theta}_k^p \in [0,1]^p$. The shared layer's parameters, meant for aggregation, enhance the model by leveraging collective insights, while the private layer's parameters are tailored for local data specificity and privacy, hence not shared. Based on this probability mask, the binary mask $\mathbf{m}_k = \text{Bern}(\boldsymbol{\theta}_k)$ inherits the bifurcated structure of $\boldsymbol{\theta}_k$, comprising shared $\mathbf{m}_k^s$ and private $\mathbf{m}_k^p$ layers. This structured approach ensures that while personalization is maintained through private layers, the collaborative benefits of federated learning are realized by the shared layers.

Each edge server e, where $e \in E$, is responsible for collecting binary masks $\mathbf{m}_k^s$, $k \in K_e$ (①-b). These edge servers aggregate (②-a) the masks using a Bayesian aggregation method, which allows adaptation to various data distributions and improves performance through refined model updates. Post-aggregation, the edge server computes a consolidated probability mask $\boldsymbol{\theta}_e \in [0,1]^s$ from the aggregated data and uses it to generate binary masks $\mathbf{m}_e \in \{0,1\}^s$ through a Bernoulli sampling process (②-b), which are then sent to the cloud server for further processing (②-c).

The cloud server plays a pivotal role in integrating the updates received from various edge servers. It performs the final aggregation of the binary masks $\mathbf{m}_e$ uploaded by the edge servers to form a global probability mask $\boldsymbol{\theta}_g \in [0,1]^s$ (③-a). This global mask $\boldsymbol{\theta}_g$ is then broadcast back to all clients through the edge servers (③-b), facilitating a synchronized update across the network. This mechanism ensures that the global model is continually refined and updated based on the collective learning from all participating clients.

Local Training in Client

In H-FedSN, for round t, clients initially receive a global shared probability mask $\boldsymbol{\theta}_{g,t-1}$ from the cloud server. Clients combine $\boldsymbol{\theta}_{g,t-1}$ with $\boldsymbol{\theta}_{k,t-1}^p$ to generate $\boldsymbol{\theta}_{k,t} \in [0,1]^d$, which represents the activation probabilities of the model parameters. To optimize within a broader parameter space, this probability mask is transformed into a *score mask* $\mathbf{s} = (s^1, s^2, \ldots, s^d) \in \mathbb{R}^d$ via the inverse Sigmoid function. This

transformation allows the *score mask* $\mathbf{s}$ to vary freely within the real number range, providing more flexibility for subsequent optimization. This design permits the *score mask* to flexibly generate the corresponding probability mask by setting $\boldsymbol{\theta} = \text{Sigmoid}(\mathbf{s})$, ensuring each value is appropriately within the [0, 1] interval. The detailed steps for local probability mask training in round t are described as follows and can also be seen in lines 8-21 in Algorithm 1.

1. Clients receive the global probability mask $\boldsymbol{\theta}_{g,t-1}$ from the cloud server, then combine $\boldsymbol{\theta}_{g,t-1}$ with $\boldsymbol{\theta}^p_{k,t-1}$ to generate $\boldsymbol{\theta}_{k,t}$. Each client then inputs this probability mask into the inverse Sigmoid function $\text{Sigmoid}^{-1}(\cdot)$, obtaining the *score mask* $\mathbf{s}_{k,t} = \text{Sigmoid}^{-1}(\boldsymbol{\theta}_{k,t})$.

2. Clients transform back to $\boldsymbol{\theta}_{k,t} = \text{Sigmoid}(\mathbf{s}_{k,t})$, then use Bernoulli sampling to extract binary masks $\mathbf{m}_{k,t}$ from $\boldsymbol{\theta}_{k,t}$.

3. The sampled binary mask $\mathbf{m}_{k,t}$ is used to sparsify the initial weight vector $\mathbf{w}_{\text{init}}$: $\dot{w}_{k,t} = \mathbf{m}_{k,t} \odot \mathbf{w}_{\text{init}}$.

4. The $\dot{w}_{k,t}$ is utilized for forward propagation to compute the outputs, and the computed local loss $L(f(\dot{w}_{k,t}), D_k)$ is then used in backpropagation to update the *score mask* according to the formula $\mathbf{s}_{k,t} = \mathbf{s}_{k,t} - \eta \nabla L(f(\dot{w}_{k,t}), D_k)$(where η is the local learning rate).

Fig. 2 illustrates the process of a single epoch of local training on a client. Steps 2 to 4 involve all local operations that are differentiable, except for the Bernoulli sampling step. Since Bernoulli sampling itself is non-differentiable, this would pose a problem in traditional gradient descent frameworks. To address this, we adopt the straight-through estimator technique proposed by [43]. This technique allows us to "pass-through" gradients during the Bernoulli sampling process: during backpropagation, we set the output gradient of the Bernoulli function to be equal to the input probability mask $\boldsymbol{\theta}_{k,t}$. Thus, although the sampling step is not differentiable, we can still effectively propagate gradients throughout the network and update the score mask $\mathbf{s}_{k,t}$. After τ training rounds, each client completes the training of their local probability mask $\boldsymbol{\theta}_{k,t} = \text{Sigmoid}(\mathbf{s}_{k,t})$ and generates a binary mask $\mathbf{m}_{k,t} = \textbf{Bern}(\boldsymbol{\theta}_{k,t})$ through Bernoulli sampling. The client keeps the private part, $\mathbf{m}^p_{k,t}$, and uploads the shared layers, $\mathbf{m}^s_{k,t}$, to the respective connected edge server. The next step involves the edge server aggregating these binary masks to form a comprehensive model update.

Bayesian Aggregation in HFL

In the HFL architecture, both edge and cloud servers implement a Bayesian aggregation strategy to process the binary masks they receive. This approach marks a significant departure from traditional federated learning methods, such as FedAvg, which primarily rely on simple averaging to aggregate updates. Unlike these conventional methods, H-FedSN leverages Bayesian aggregation to more effectively account for the inherent uncertainties in data updates. This capability allows H-FedSN to construct more robust and adaptive global models by synthesizing updates represented as binary masks from clients across multiple layers of the architecture. The subsequent sections provide a detailed description of the aggregation processes at both the edge server and cloud server levels.

Fig. 3 shows that each edge server $e \in E$ collects binary masks $\mathbf{m}_{k,t}^s$ from its directly connected clients $k \in K_e$ at round t and aggregates them into an edge-level probability mask $\boldsymbol{\theta}_{e,t}$ using Bayesian aggregation (Algorithm2). Specifically, $\boldsymbol{\theta}_{e,t}$ is modeled by a Beta distribution with parameters $\alpha_{e,t}$ and $\beta_{e,t}$, both initialized as $\lambda_0 = 1$ at round $t = 0$.

At round $t = 0$, the system has no prior knowledge, the probability mask follows a uniform distribution over [0, 1]. Upon receiving client masks $\mathbf{m}_{k,t}^s$ from client $k \in K_e$, edge e updates the prior to a posterior. Exploiting the conjugacy of Beta and Bernoulli distributions, the posterior remains Beta-distributed with updated parameters: Since Beta and Bernoulli distributions form a conjugate pair, the posterior conveniently remains a Beta distribution with updated parameters: $\alpha_{e,t} = \alpha_{e,t-1} + \sum_{k \in K_e} \mathbf{m}_{k,t}^s$ and $\beta_{e,t} = \beta_{e,t-1} + |K_e| \cdot \mathbf{1} - \sum_{k \in K_e} \mathbf{m}_{k,t}^s$, where $\mathbf{m}_{k,t}^s$ represents binary mask submitted by the k-th client under edge server e during round t. The $\mathbf{1}$ is the s-dimensional all-ones vector. $|K_e|$ denotes the number of clients connected to edge server e. Subsequently, the edge server calculates the probability mask $\boldsymbol{\theta}_{e,t}$ based on the updated parameters $\alpha_{e,t}$ and $\beta_{e,t}$: $\boldsymbol{\theta}_{e,t} = \frac{\alpha_{e,t}-1}{\alpha_{e,t}+\beta_{e,t}-2}$. This operation is applied element-wise. The edge server then broadcasts $\mathbf{m}_{e,t} = \text{Bern}(\boldsymbol{\theta}_{e,t})$ to the cloud. To avoid overfitting the model to historical data, $\alpha_{e,t}$ and $\beta_{e,t}$ are reset to their initial value $\lambda_0 = 1$ at the start of each 10^{th} training round. This historical window size w provides a practical frequency for incorporating recent updates while preventing older statistics from dominating indefinitely. We provide an ablation on w in Supplementary Table 2. For the pseudocode of edge server Bayesian aggregation, see Algorithm 2.

Algorithm 2 Bayesian Aggregation at Edge

Require: Binary masks $\mathbf{m}_{k,t}^s$ from clients $k \in K_e$, round number t, reset interval w
Ensure: Updated binary mask $\boldsymbol{m}_{e,t}$

if $t = 0$ **or** $t \bmod w = 0$ **then**
    $\alpha_{e,t-1} = \beta_{e,t-1} = \lambda_0$ {reset Beta priors}
end if
$\alpha_{e,t} = \alpha_{e,t-1} + \sum_{k \in K_e} \mathbf{m}_{k,t}^s$
$\beta_{e,t} = \beta_{e,t-1} + |K_e| \cdot \mathbf{1} - \sum_{k \in K_e} \mathbf{m}_{k,t}^s$
$\boldsymbol{\theta}_{e,t} = \frac{\alpha_{e,t}-1}{\alpha_{e,t}+\beta_{e,t}-2}$
$\boldsymbol{m}_{e,t} \leftarrow \textbf{Bern}(\boldsymbol{\theta}_{e,t})$
Output: $\boldsymbol{m}_{e,t}$

The cloud server is responsible for collecting the binary masks uploaded by all edge servers e, $e \in E$, and performs a global aggregation.

At the cloud level, as illustrated in Fig. 4 and Algorithm 3, the global probability mask $\boldsymbol{\theta}_{g,t}$ is modeled using a Beta distribution Beta$(\alpha_{g,t}, \beta_{g,t})$, with parameters $\alpha_{g,t}$ and $\beta_{g,t}$ related to the training round and initially set to $\alpha_{g,0} = \beta_{g,0} = \lambda_0$. The parameters for the updated posterior distribution are calculated as follows: $\alpha_{g,t} = \alpha_{g,t-1} + \sum_{e \in E} \mathbf{m}_{e,t}$ and $\beta_{g,t} = \beta_{g,t-1} + |E| \cdot \mathbf{1} - \sum_{e \in E} \mathbf{m}_{e,t}$

, where $\mathbf{m}_{e,t}$ represents the binary mask submitted by the e-th edge server to the cloud during round t. The $\mathbf{1}$ is the s-dimensional all-ones vector. $|E|$ denotes the number of edge servers. Following the parameter update, the cloud server computes the mode of the Bernoulli distributions to derive the global probability mask $\boldsymbol{\theta}_{g,t}$, as suggested by [44]: $\boldsymbol{\theta}_{g,t} = \frac{\alpha_{g,t}-1}{\alpha_{g,t}+\beta_{g,t}-2}$.

Subsequently, the cloud broadcasts the probability mask $\boldsymbol{\theta}_{g,t}$ to all clients through the edge servers. To avoid overfitting the model to historical data, $\alpha_{g,t}$ and $\beta_{g,t}$ are reset to their initial value $\lambda_0 = 1$ at the start of each 10^{th} training round, which provides a practical frequency for incorporating recent updates while preventing older statistics from dominating indefinitely. For the pseudocode of cloud server Bayesian aggregation, see Algorithm 3.

Algorithm 3 Bayesian Aggregation at Cloud

Require: Binary masks $\mathbf{m}_{e,t}$ from edge servers $e \in E$, round number t, reset interval w
Ensure: Updated global probability mask $\boldsymbol{\theta}_{g,t}$
if $t = 0$ **or** $t \bmod w = 0$ **then**
    $\alpha_{g,t-1} = \beta_{g,t-1} = \lambda_0$ {reset Beta priors}
end if
$\alpha_{g,t} = \alpha_{g,t-1} + \sum_{e \in E} \mathbf{m}_{e,t}$
$\beta_{g,t} = \beta_{g,t-1} + |E| \cdot \mathbf{1} - \sum_{e \in E} \mathbf{m}_{e,t}$
$\boldsymbol{\theta}_{g,t} = \frac{\alpha_{g,t}-1}{\alpha_{g,t}+\beta_{g,t}-2}$
Output: $\boldsymbol{\theta}_{g,t}$

Data availability

All datasets analysed in this study are publicly available benchmark datasets (MNIST, WIDAR, and WISDM WATCH/PHONE), and no new data were generated.

Code availability

The code implementing H-FedSN is publicly available at https://github.com/hfedsn/H-FedSN.

Acknowledgments

This research was funded by Stanford's Center for Sustainable Development and Global Competitiveness (SDGC) and the Yonghua Foundation. The funders had no role in study design, data collection and analysis, the decision to publish, or preparation of the manuscript.

Author contributions

Author contributions: conceptualization, methodology, and writing – original draft preparation: J.G. and Y.L.; writing – review and editing: Y.Z. and J.W.; figure preparation: Y.L.; supervision and project administration: B.C. and M.L. All authors reviewed and approved the final manuscript.

Competing Interests

The authors declare no competing financial or non-financial interests.

References

[1] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

[2] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[3] Latif U. Khan, Walid Saad, Zhu Han, Ekram Hossain, and Choong Seon Hong. Federated learning for internet of things: Recent advances, taxonomy, and open challenges. *IEEE Communications Surveys & Tutorials*, 23(3):1759–1799, 2021.

[4] Tuo Zhang, Lei Gao, Chaoyang He, Mi Zhang, Bhaskar Krishnamachari, and A Salman Avestimehr. Federated learning for the internet of things: Applications, challenges, and opportunities. *IEEE Internet of Things Magazine*, 5(1):24–29, 2022.

[5] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016.

[6] Mahadev Satyanarayanan. The emergence of edge computing. *Computer*, 50(1):30–39, 2017.

[7] Leonidas G Anthopoulos. Understanding the smart city domain: A literature review. *Transforming city governments for successful smart cities*, pages 9–21, 2015.

[8] A. L. Virk, Mehmood Ali Noor, S. Fiaz, S. Hussain, H. Hussain, M. Rehman, M. Ahsan, and Wei Ma. Smart farming: An overview. *Smart Village Technology*, 2020.

[9] Vu Khanh Quy, Nguyen Van Hau, Dang Van Anh, and Le Anh Ngoc. Smart healthcare iot applications based on fog computing: architecture, applications and challenges. *Complex & Intelligent Systems*, 8(5):3805–3815, 2022.

[10] Lumin Liu, Jun Zhang, SH Song, and Khaled B Letaief. Client-edge-cloud hierarchical federated learning. In *ICC 2020-2020 IEEE international conference on communications (ICC)*, pages 1–6. IEEE, 2020.

[11] Saheed A. Tijani, Xingjun Ma, Ran Zhang, Frank Jiang, and Robin Doss. Federated learning with extreme label skew: A data extension approach. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2021.

[12] Jianhang Sheng, Jian Xiong, and Bo Liu. Federated learning technology in serial topology for iot networks. *2023 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pages 1–5, 2023.

[13] Junkang Liu, Fanhua Shang, Yuanyuan Liu, Hongying Liu, Yuangang Li, and YunXiang Gong. FedBCGD: Communication-efficient accelerated block coordinate gradient descent for federated learning. In *ACM Multimedia 2024*, 2024.

[14] Yu Sun, Xin Gao, Ying Shen, Jianfei Xie, Jiandong Yang, and Nong Si. A model parameter update strategy for enhanced asynchronous federated learning algorithm. In *2023 IEEE 3rd International Conference on Computer Systems (ICCS)*, pages 9–15, 2023.

[15] Jinjin Xu, Wenli Du, Yaochu Jin, Wangli He, and Ran Cheng. Ternary compression for communication-efficient federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 33(3):1162–1176, 2022.

[16] Xin-Chun Li and De-Chuan Zhan. Fedrs: Federated learning with restricted softmax for label distribution non-iid data. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 995–1005, New York, NY, USA, 2021. Association for Computing Machinery.

[17] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning: A meta-learning approach. *arXiv preprint arXiv:2002.07948*, 2020.

[18] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818*, 2019.

[19] Chunmei Ma, Xiangqian Li, Baogui Huang, Guangshun Li, and Fengyin Li. Personalized client-edge-cloud hierarchical federated learning in mobile edge computing. *Journal of Cloud Computing*, 13(1), December 2024.

[20] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[21] Gary M Weiss, Kenichi Yoneda, and Thaier Hayajneh. Smartphone and smartwatch-based biometrics using activities of daily living. *Ieee Access*, 7:133190–133202, 2019.

[22] Zheng Yang, Yi Zhang, Guidong Zhang, Yue Zheng, and Guoxuan Chi. Widar 3.0: Wifi-based activity recognition dataset, 2020.

[23] Yue Zheng, Yi Zhang, Kun Qian, Guidong Zhang, Yunhao Liu, Chenshu Wu, and Zheng Yang. Zero-effort cross-domain gesture recognition with wi-fi. In *Proceedings of the 17th annual international conference on mobile systems, applications, and services*, pages 313–325, 2019.

[24] Jeffrey W Lockhart, Gary M Weiss, Jack C Xue, Shaun T Gallagher, Andrew B Grosner, and Tony T Pulickal. Design considerations for the wisdm smart phone-based sensor mining architecture. In *Proceedings of the Fifth International Workshop on Knowledge Discovery from Sensor Data*, pages 25–33, 2011.

[25] Yujia Wang, Lu Lin, and Jinghui Chen. Communication-efficient adaptive federated learning. *ArXiv*, abs/2205.02719, 2022.

[26] Alham Fikri Aji and Kenneth Heafield. Sparse communication for distributed gradient descent. *arXiv preprint arXiv:1704.05021*, 2017.

[27] Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, and Michael L Littman. Activity recognition from accelerometer data. In *Aaai*, volume 5, pages 1541–1546. Pittsburgh, PA, 2005.

[28] Jorge-L Reyes-Ortiz, Luca Oneto, Albert Samà, Xavier Parra, and Davide Anguita. Transition-aware human activity recognition using smartphones. *Neurocomputing*, 171:754–767, 2016.

[29] Charissa Ann Ronao and Sung-Bae Cho. Human activity recognition with smartphone sensors using deep learning neural networks. *Expert systems with applications*, 59:235–244, 2016.

[30] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. In *2022 IEEE 38th international conference on data engineering (ICDE)*, pages 965–978. IEEE, 2022.

[31] Santanu Purohit et al. Evaluation of the efficacy of iot deployment on petro-retail operations. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, 12(4):356–363, 2021.

[32] Hoa Hong Nguyen, Farhaan Mirza, M. Asif Naeem, and Minh Nguyen. A review on iot healthcare monitoring applications and a vision for transforming sensor data into real-time clinical feedback. In *2017 IEEE 21st International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 257–262, 2017.

[33] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.

[34] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*, pages 5132–5143. PMLR, 2020.

[35] Ahmed Imteaj, Urmish Thakker, Shiqiang Wang, Jian Li, and M. Amini. A survey on federated learning for resource-constrained iot devices. *IEEE Internet of Things Journal*, 9:1–24, 2021.

[36] Wanli Ni, Jingheng Zheng, and Hui Tian. Semi-federated learning for collaborative intelligence in massive iot networks. *IEEE Internet of Things Journal*, 10:11942–11943, 2023.

[37] L. U. Khan, W. Saad, Zhu Han, E. Hossain, and C. Hong. Federated learning for internet of things: Recent advances, taxonomy, and open challenges. *IEEE Communications Surveys & Tutorials*, 23:1759–1799, 2020.

[38] Qiong Wu, Kaiwen He, and Xu Chen. Personalized federated learning for intelligent iot applications: A cloud-edge based framework. *IEEE Open Journal of the Computer Society*, 2020.

[39] Xueyu Wu, Xin Yao, and Cho-Li Wang. Fedscr: Structure-based communication reduction for federated learning. *IEEE Transactions on Parallel and Distributed Systems*, 32(7):1565–1577, 2021.

[40] M. Nori, Sangseok Yun, and Il Kim. Fast federated learning by balancing communication trade-offs. *IEEE Transactions on Communications*, 69:5168–5182, 2021.

[41] Jian ji Ren, Wanli Ni, and Hui Tian. Toward communication-learning trade-off for federated learning at the network edge. *IEEE Communications Letters*, 26:1858–1862, 2022.

[42] Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. Deconstructing lottery tickets: Zeros, signs, and the supermask. *Advances in neural information processing systems*, 32, 2019.

[43] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[44] Paulo Abelha Ferreira, Pablo Nascimento da Silva, Vinicius Gottin, Roberto Stelling, and Tiago Calmon. Bayesian signsgd optimizer for federated learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Figure legends

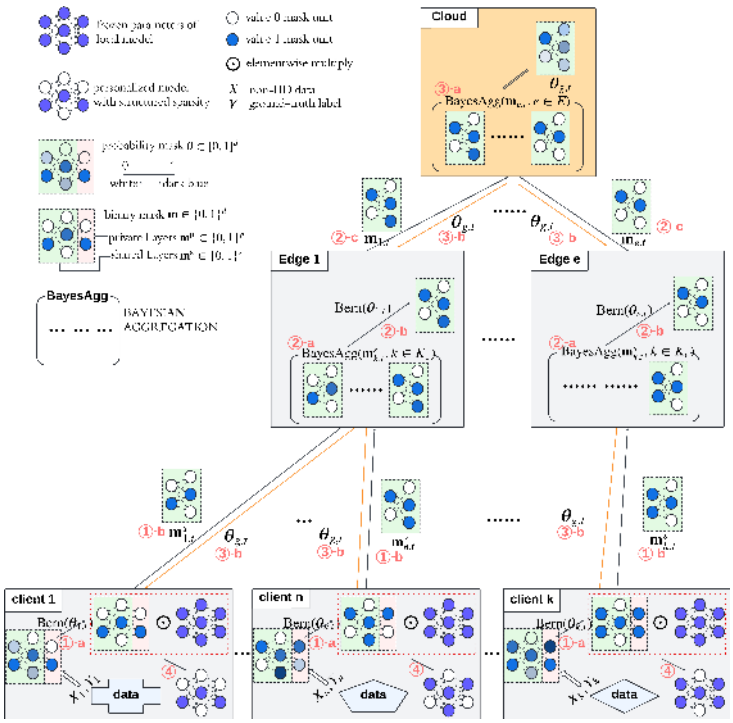

Figure 1: **Overview of the H-FedSN framework across the client, edge and cloud tiers of a hierarchical federated learning (HFL) system for Internet-of-Things (IoT) devices.** Throughout, filled blue circles denote mask units with value 1 and open white circles denote mask units with value 0. Filled purple circles denote the frozen parameters of the randomly initialized local model, and a circled dot denotes element-wise multiplication. The probability mask $\boldsymbol{\theta} \in [0,1]^d$ uses a colour scale from white (0) to dark blue (1). The binary mask $\mathbf{m} \in \{0,1\}^d$ has shared layers $\mathbf{m}^s$ (green shading) and private, personalized layers $\mathbf{m}^p$ (pink shading). At each client (bottom), a binary mask is sampled as $\mathrm{Bern}(\boldsymbol{\theta}_{k,t})$ and applied to the frozen weights, using the client's non-IID data X and labels Y. Each edge server (middle) performs Bayesian aggregation (BayesAgg) of the shared client masks $\mathbf{m}^s_{k,t}$ over its clients $k \in K_e$, then samples and uploads an edge mask $\mathbf{m}_{e,t}$. The cloud server (top, orange box) aggregates the edge masks over all edges $e \in E$ into the global probability mask $\boldsymbol{\theta}_{g,t}$, which is broadcast back to the clients through the edges (orange lines). Green-bordered boxes group the units of a mask, and the dots denote additional clients or edges. Encircled numbers 1–4, with sub-labels a, b and c, give the order of operations.

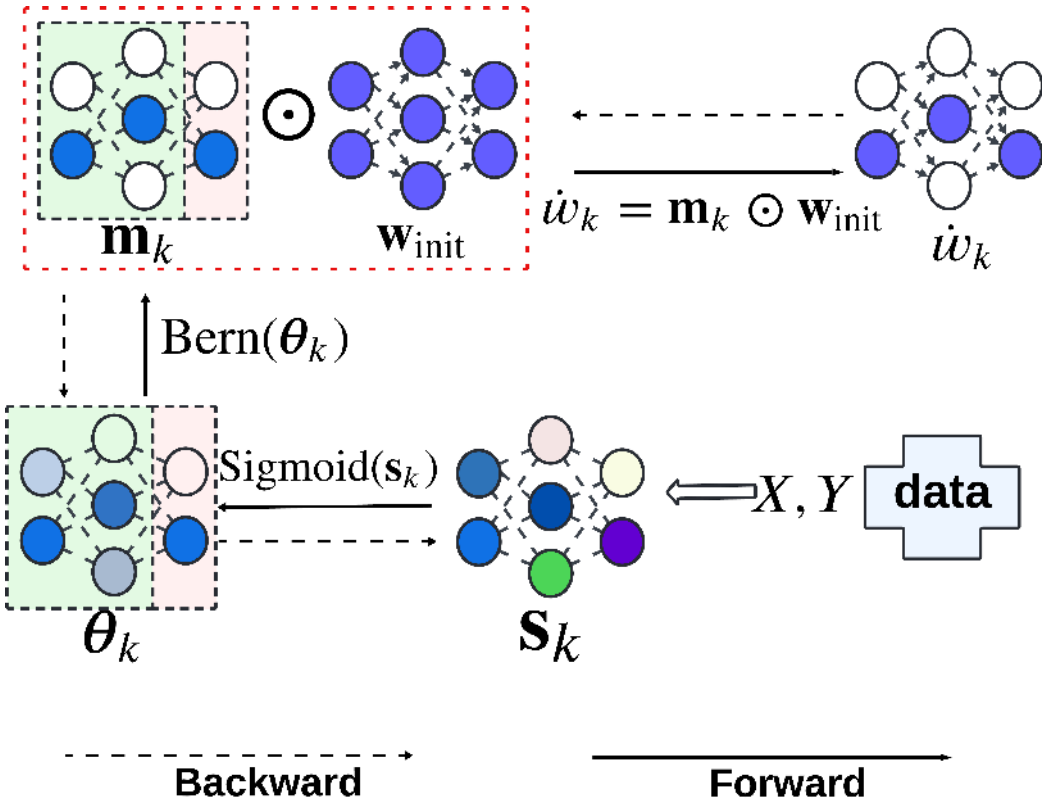

Figure 2: **H-FedSN Local Training in Client.** The schematic traces a single forward and backward pass through a frozen, randomly initialized network whose binary mask is the only trainable object. The forward pass uses solid arrows, defined in the lower legend. From the client's local data, input features X and labels Y, the client forms a real-valued score mask $\mathbf{s}_k$ (centre), whose circles take varied colours to denote unconstrained real-valued scores. A sigmoid maps these scores to a probability mask $\boldsymbol{\theta}_k \in [0,1]^d$ (lower left), shaded from pale (probability near 0) to dark blue (probability near 1). Bernoulli sampling, $\mathrm{Bern}(\boldsymbol{\theta}_k)$, then draws a binary mask $\mathbf{m}_k \in \{0,1\}^d$ in which filled blue circles are units with value 1 and white circles are value 0 (upper left). Element-wise multiplication (circled dot) with the frozen initial weights $\mathbf{w}_{\text{init}}$ (uniformly filled purple circles) yields the sparse model $\dot{w}_k = \mathbf{m}_k \odot \mathbf{w}_{\text{init}}$ (upper right); the red dashed rectangle encloses this masking step. In both $\mathbf{m}_k$ and $\boldsymbol{\theta}_k$, light-green shading marks the shared layers and light-pink shading the private (personalized) layers. The backward pass uses dashed arrows: gradients propagate from the sparse model back to $\boldsymbol{\theta}_k$ and on to the score mask $\mathbf{s}_k$. Because Bernoulli sampling is non-differentiable, these gradients pass through it by the straight-through estimator.

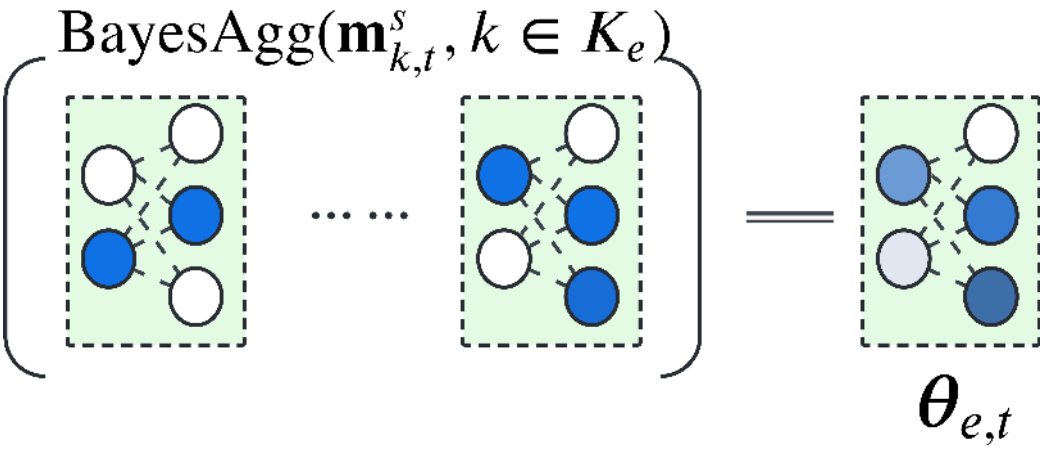

Figure 3: **Bayesian aggregation of client masks at an edge server in H-FedSN.** The figure shows how one edge server e combines the shared binary masks $\mathbf{m}^s_{k,t}$ that its connected clients $k \in K_e$ upload at round t (left) into a single edge-level probability mask $\boldsymbol{\theta}_{e,t}$ (right), written $\boldsymbol{\theta}_{e,t} = \mathrm{BayesAgg}(\mathbf{m}^s_{k,t},\, k \in K_e)$. Each client mask is a small network in which filled blue circles are units with value 1 and white circles are value 0.

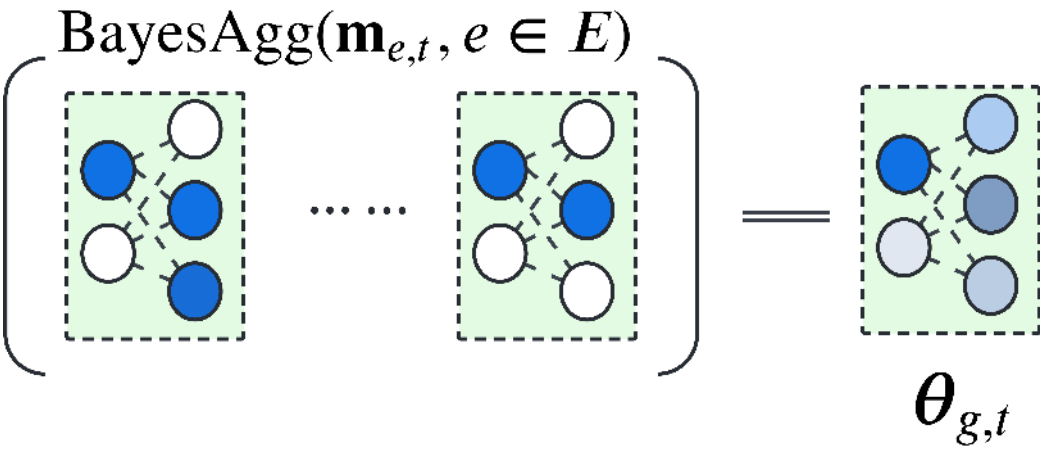

Figure 4: **Bayesian aggregation of edge masks at the cloud server in H-FedSN.** The cloud server combines the binary masks $\mathbf{m}_{e,t}$ that all edge servers $e \in E$ upload into the global probability mask $\boldsymbol{\theta}_{g,t}$ by Beta–Bernoulli Bayesian aggregation, written above the bracketed term as $\mathrm{BayesAgg}(\mathbf{m}_{e,t},\, e \in E)$. The cloud then broadcasts $\boldsymbol{\theta}_{g,t}$ to all clients through the edge servers. BayesAgg, Bayesian aggregation.

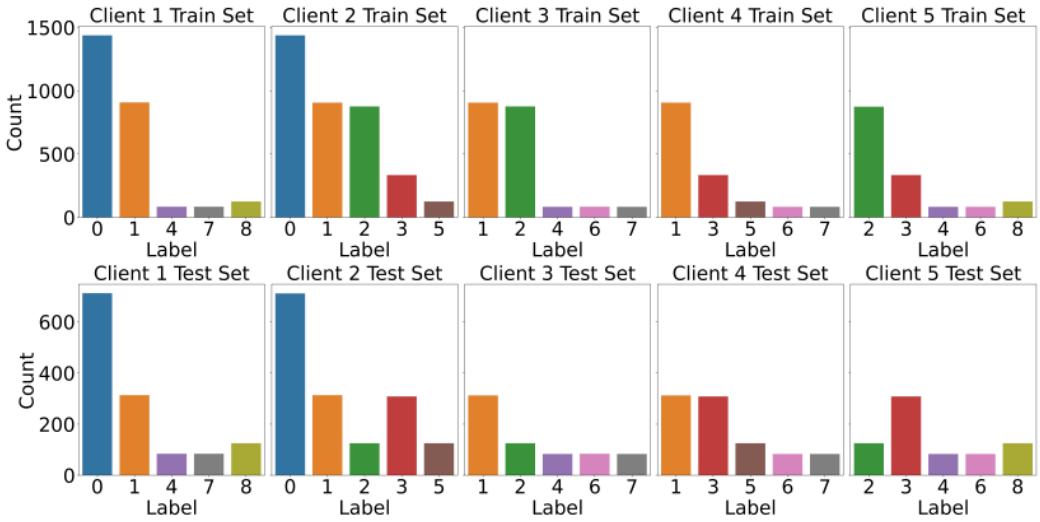

Figure 5: **WIDAR Dataset Non-IID Data Distribution Across 5 Clients.** Each panel is a bar chart of the per-class sample count for one client, with the class label (0–8) on the horizontal axis and the sample count on the vertical axis. The top row gives the training-set distributions for clients 1–5, and the bottom row gives the matching test-set distributions. Bars are coloured by class label: class 0 blue, class 1 orange, class 2 green, class 3 red, class 4 purple, class 5 brown, class 6 pink, class 7 grey and class 8 olive. Each client holds samples from only a fixed subset of the classes, and within a client the per-class counts are imbalanced.

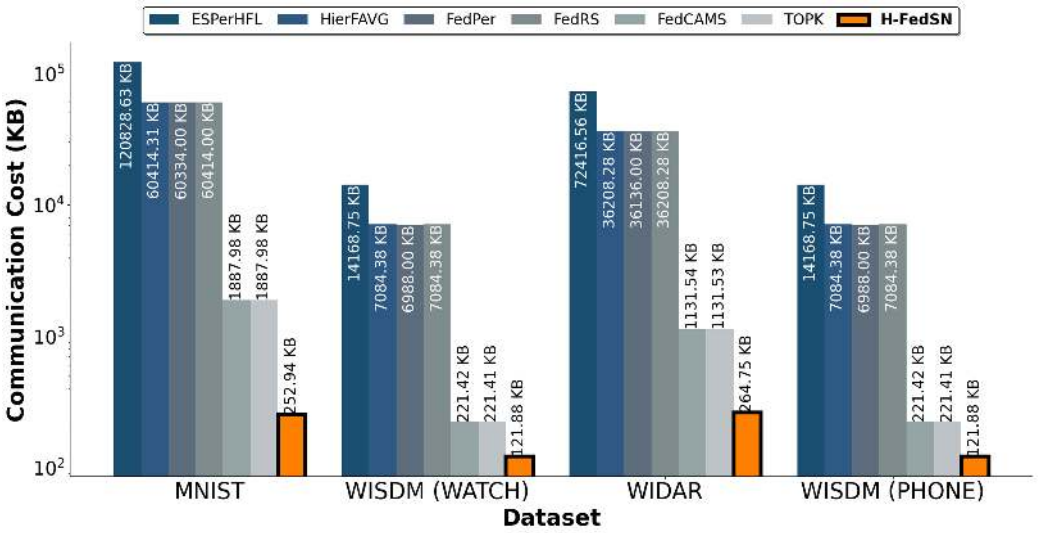

Figure 6: **Communication cost for different algorithms across different datasets in all settings.** The grouped bar chart reports the per-round communication cost in kilobytes (KB), with a base-10 logarithmic vertical axis and bars grouped by dataset along the horizontal axis (MNIST, WISDM (WATCH), WIDAR and WISDM (PHONE)). Within each group the bars run, from left to right, ESPerHFL, HierFAVG, FedPer, FedRS, FedCAMS, TOPK and H-FedSN, in the order of the colour key above the plot.

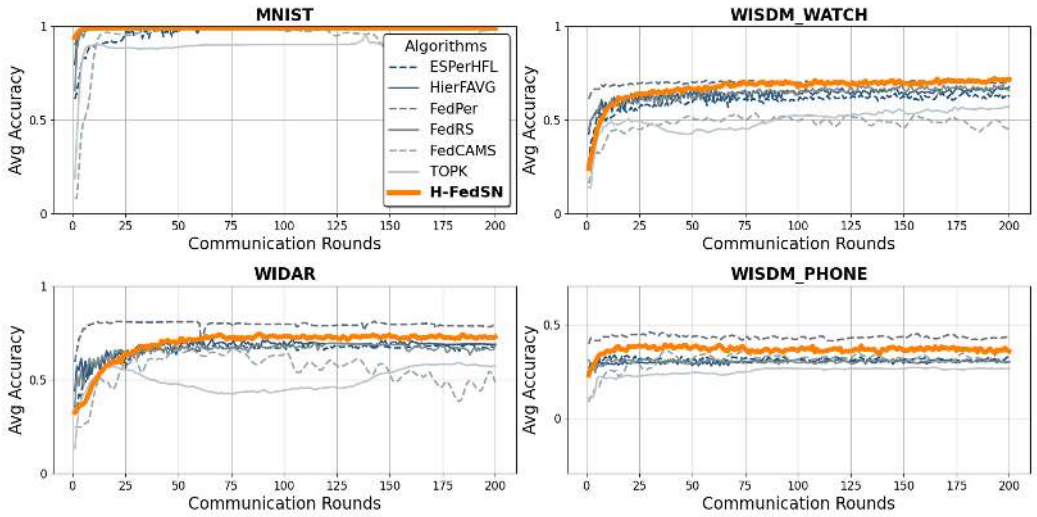

Figure 7: **Average client accuracy versus communication round under the E2C5 setting across four datasets.** Each panel plots the mean inference accuracy over all clients (vertical axis, "Avg Accuracy") against the communication round (horizontal axis, "Communication Rounds", 0 to 200) for one dataset.

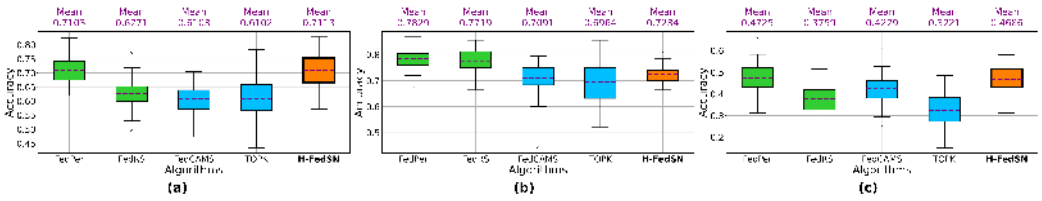

Figure 8: **Distribution of per-client inference accuracy under the imbalanced E5C50 configuration.** The three panels are box plots of per-client accuracy for each method on (a) WISDM (WATCH), (b) WIDAR and (c) WISDM (PHONE). In every panel the methods, from left to right, are FedPer, FedRS, FedCAMS, TOPK and H-FedSN. Box fill colour marks the method family: green for the personalization methods FedPer and FedRS, cyan for the communication-efficient methods FedCAMS and TOPK, and orange with a bold black outline for H-FedSN.

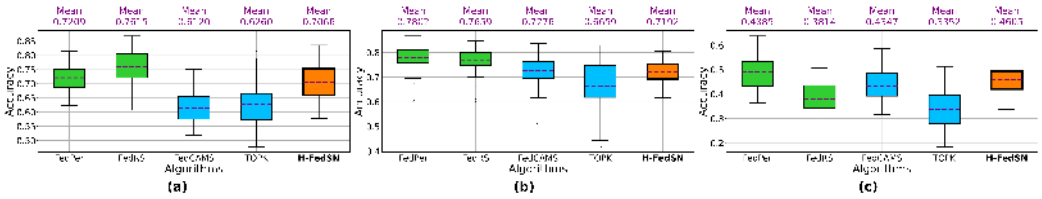

Figure 9: **Distribution of per-client inference accuracy under the E20C50 configuration.** The three panels are box plots of per-client accuracy for each method on (a) WISDM (WATCH), (b) WIDAR and (c) WISDM (PHONE). In every panel the methods, from left to right, are FedPer, FedRS, FedCAMS, TOPK and H-FedSN. Box fill colour marks the method family: green for the personalization methods FedPer and FedRS, cyan for the communication-efficient methods FedCAMS and TOPK, and orange with a bold black outline for H-FedSN.

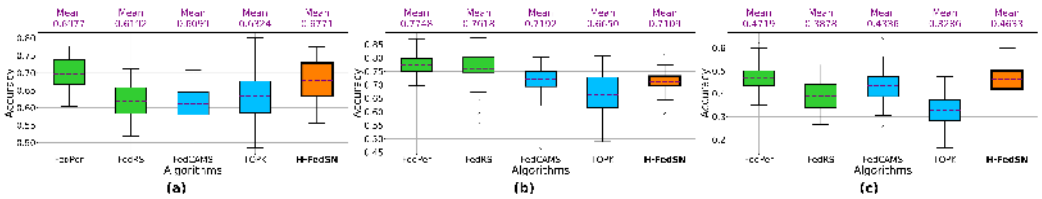

Figure 10: **Distribution of per-client inference accuracy under the E2C50 configuration.** The three panels are box plots of per-client accuracy for each method on (a) WISDM (WATCH), (b) WIDAR and (c) WISDM (PHONE). In every panel the methods, from left to right, are FedPer, FedRS, FedCAMS, TOPK and H-FedSN. Box fill colour marks the method family: green for the personalization methods FedPer and FedRS, cyan for the communication-efficient methods FedCAMS and TOPK, and orange with a bold black outline for H-FedSN.