

**Efficient Hierarchical Federated Learning with Dynamic Cluster  
Formation for IoT Wearable Analytics**

**Senanayake, S., Villarini, B., Kiss, T. and Tusa, F.**

This is a copy of the author's accepted version of a paper subsequently to be published in the proceedings of DCOSS-IoT 2026 – 22nd Conference on Distributed Computing in Smart Systems and the Internet of Things, Reykjavik, 22 - 24 Jun 2026.

The final published version will be available online at:

<https://doi.org/10.1109/DCOSS-IoT69657.2026.00002>

© 2026 IEEE . This manuscript version is made available under the CC-BY 4.0 license

<https://creativecommons.org/licenses/by/4.0/>

For the purpose of open access, the author(s) has applied a Creative Commons Attribution (CC BY) license to any Accepted Manuscript version arising.

The WestminsterResearch online digital archive at the University of Westminster aims to make the research output of the University available to a wider audience. Copyright and Moral Rights remain with the authors and/or copyright owners.

# Efficient Hierarchical Federated Learning with Dynamic Cluster Formation for IoT Wearable Analytics

Nipuna Senanayake

*Department of Computer Science and Engineering  
University of Westminster  
London, United Kingdom  
w2030483@westminster.ac.uk*

Tamas Kiss

*Department of Computer Science and Engineering  
University of Westminster  
London, United Kingdom  
t.kiss@westminster.ac.uk*

Barbara Villarini

*Department of Computer Science and Engineering  
University of Westminster  
London, United Kingdom  
b.villarini@westminster.ac.uk*

Francesco Tusa

*Department of Computer Science and Engineering  
University of Westminster  
London, United Kingdom  
f.tusa@westminster.ac.uk*

**Abstract**— Machine learning (ML) models generally require large datasets for better accuracy. However, data privacy and security concerns impose constraints on centralized data collection and model training. Federated Learning (FL) addresses this by training models without data leaving its origin, but FL still encounters challenges regarding training efficiency, model convergence and model accuracy due to data heterogeneity among clients. This work proposes a Hierarchical Federated Learning (HFL) framework that introduces an intermediate aggregation layer that dynamically groups similar clients into clusters, allowing analysis of privacy and accuracy trade-off through two alternative approaches: HFL+NoDataMove (within the cluster model weight aggregation for better privacy) and HFL+DataMove (within the cluster data pooling for improved accuracy). Both alternatives employ Locality Sensitive Hashing (LSH) for lightweight cluster building and optionally quantization for communication efficiency. Our experiments to evaluate the proposed framework were conducted on a real-life Fitbit activity dataset of 33 users using linear regression as the model for prediction of calories burnt during daily activity. Experiments across 10 independent experiment runs demonstrate that HFL+DataMove reaches centralized accuracy (mean MSEs of 0.070 vs 0.057, respectively), significantly outperforming three FL benchmarks FedAvg, FedProx, SCAFFOLD, and one proposed method HFL+NoDataMove ( $p < 0.001$ , Cohen's  $d > 2.5$ ). Experiments were conducted to evaluate the proposed framework in terms of model performance, communication overhead, and computational cost, with Mean Squared Error used as the primary performance metric. The results suggest that the proposed methodology is feasible for IoT (Internet of Things) deployment, with total communication overhead remaining below 400 KB across 10 global rounds.

**Keywords**—*Hierarchical Federated Learning, Locality Sensitive Hashing, Quantization, Data Movement, IoT Wearables*

## I. INTRODUCTION

IoT wearable devices have become increasingly common in the personal health sector, reflecting their growing popularity across various fields. Examples include fitness trackers, smartwatches, and other wearable devices, which all offer the opportunity to collect sensitive physiological data from millions of users worldwide. While this data availability is strongly in support of better models, it may lead to several challenges on the other hand, because centralizing such

personal data can raise significant privacy, regulatory, and security concerns. It is evident that if we use all user data to train a single model at a central location, the resulting model will achieve higher performance than multiple models trained in a distributed setup with the focus of privacy preservation. In the context of Industry 5.0 [1], the importance of human-centric, innovation and technology that serves human wellbeing is in demand (as opposed to traditional goals like mass production and automation). This enables users to retain control over their personal data while still benefiting from collaborative intelligence. Also, it aligns well with Federated Learning (FL) [2], which provides a technical foundation for implementing this principle.

FL is a distributed ML technique where several individual clients with similar types of data can get together and build aggregated models without sharing the data itself. By aggregating locally trained updates across many clients, FL preserves data privacy while still leveraging the collective learning signal from the entire client population offering better potential accuracy than any single client could achieve alone. In a world where privacy is a concern, there is always a trade-off between model accuracy and user data privacy. FL in its naïve form would be one of the best choices if privacy is the top priority rather than accuracy. However, FL still encounters significant challenges. The distributed nature introduces communication overhead, and data heterogeneity across clients can impose challenges in IoT deployments due to individual devices capturing distinct behavioral patterns leading to significant accuracy degradation compared to centralized training. Standard FL methods such as FedAvg [2], FedProx [3], and SCAFFOLD [4] are some of the key benchmark algorithms that attempt to address challenges in convergence and communication, but they still report a significant accuracy gap relative to centralized training outcomes. This raises an interesting question: Can we bridge this gap while sticking to the distributed architecture?

This work proposes a Hierarchical Federated Learning (HFL) framework that introduces an intermediate aggregation layer which dynamically groups similar clients into clusters, tackling exactly the above question by allowing practitioners to prioritize one of the two conflicting outcomes of distributed ML: privacy vs accuracy. The framework employs Locality Sensitive Hashing (LSH) as a lightweight, hash-based

computational technique for grouping clients into dynamic clusters, and two operating modes: 1) HFL+NoDataMove where the privacy is prioritized over accuracy and hence within the cluster model weight aggregation is performed. 2) HFL+DataMove in which the accuracy is prioritized over privacy leading to a setup where clients transfer local data to a cluster anchor node for pooled training within trust boundaries. Optionally, both techniques use quantization for compression of model parameters between aggregation layers to reduce communication overhead. As a preliminary investigation, we evaluate this framework using a linear regression model on a real-life Fitbit wearable dataset of 33 users to demonstrate the approach and identify which components drive accuracy improvement. Generalization to deeper models is an important direction for future work.

Our contributions can be summarized as follows: we present a configurable HFL framework that allows us to prioritize one of the two conflicting outcomes of distributed ML, privacy vs accuracy in the context of diverse IoT deployments. Further, we have conducted rigorous experiments with 10 different random seed configurations that prove the statistical significance of the results, particularly using ANOVA, pairwise Welch's t-tests, and effect size analysis against five baselines including three standard federated aggregation techniques and the two proposed methods. A study for model accuracy based on the two proposed methods has been conducted to see the impact of data movement. Further to these, we analyze the communication cost vs accuracy trade-off, exploring the feasibility of our methods for IoT deployments.

## II. RELATED WORK

When efficiency improvement in FL is considered, several areas have been studied in research efforts. Communication efficiency is probably the most studied topic, branching to areas like: (a) Improving local computation efficiency; (b) Efficient resource management in client and aggregators; (c) Integration of cloud, fog and edge computing technologies in the FL process to leverage on its infrastructure to make complex computation relatively faster; (d) Security and robustness as a measure of efficiency; (e) Efficient energy management (especially when dealing with mobile and IoT systems); and (f) even expanding to Discussion of Fairness and explainability in ML models built in federated setup. Ultimately, model accuracy is itself a dimension of efficiency: a framework that reduces communication or computation at the cost of significant accuracy loss is not truly efficient. Practical FL systems must therefore balance accuracy against communication and computation costs rather than optimizing any single dimension in isolation. These directions address FL efficiency from different angles, but mostly in isolation. In contrast, the proposed framework combines hierarchical coordination with controlled within-cluster data movement.

Focusing on communication efficiency, we can reduce rounds of communication by selecting participants [5] rather than using the whole cohort in each communication round. This approach not only focuses on efficiency but also incorporates security in adversarial environments where the clients are attempting to mislead the global aggregator by sharing adversarial model updates or the other way around [6]. Model compression efficiently reduces communication overhead, which has been achieved through techniques like quantization, pruning [7] and random rotation [8]. Further to this, model updates could also be structured to reduce

communication costs using initiatives such as low-rank approximation [9] and random masking [10]. These techniques introduce a trade-off between communication efficiency and model accuracy, though recent work has shown that carefully designed compression schemes can preserve accuracy within acceptable bounds [11]. Hardware and network optimization could impact positively on FL if the techniques are utilized to overlap local computation on devices with communication with the server [12]. However, these approaches mainly reduce the cost of transmitting model updates. In contrast, our framework improves efficiency through selective data consolidation within trusted clusters before training.

In the local models, compression and sparsification [13] has been employed to reduce the model complexity which not only speeds up local computation but also reduces the model parameters being transferred through the network and hence reduces the communication cost. Standard optimization algorithms would not perform well in local model optimization [14]. So, alternative algorithms have to be explored for this purpose. Unlike methods that optimize or compress local models alone, our approach reduces redundant training through pooled learning within hierarchical clusters.

With local device resource management in mind, several attempts have been launched focusing computation power and memory across participating devices based on device capabilities, data distribution, and workload balancing [15, 16]. Efficient caching mechanisms at device level would also benefit significantly in FL environments. Cloud and Fog technologies have contributed in improving the FL efficiency in numerous ways including: Two-layer architecture with fog nodes handling local training on devices [17] and cloud servers performing global aggregation and model updates, Fog assisted secure aggregation [18] to improve the security. Use of fog computing for collaborative learning [19], Addressing the challenges of resource limitations on mobile devices by leveraging edge cloud offloading [20] etc. These works demonstrate that introducing an intermediate aggregation tier between clients and the global server can reduce communication load and improve training efficiency. Inspired by these, we introduce a HFL framework that builds on and extends through dynamic similarity-based clustering rather than static fog-node assignment.

In terms of model aggregation, FedProx [3] extends FedAvg [2] by adding a proximal regularization term that limits how far local models can drift from the global model. SCAFFOLD addresses client drift through control variates that correct the local update direction. When the architecture of distributed FL models is looked into, Hierarchical FL architectures introduce intermediate aggregation layers. [17] proposed a three-tier architecture spanning cloud, edge, and end devices. [18] developed fog-assisted FL for resource-constrained IoT devices. These works demonstrate the benefits of hierarchical organization but typically rely on weight aggregation alone without exploring within-cluster data consolidation. Unlike, these methods, our framework combines hierarchical FL with within-cluster data movement, extending aggregation beyond weight exchange alone.

It is interesting to note that the privacy guarantees of standard FL are more sophisticated than they are generally assumed to be. Gradient inversion attacks [21] have demonstrated that shared model weights can be used to reconstruct individual training examples, making the

distinction between sharing weights and sharing data quantitative rather than qualitative. This observation motivates treating the privacy trade-off as explicit and configurable, which is reflected in our two proposed approaches: HFL+NoDataMove retains standard FL privacy by sharing only model weights, and HFL+DataMove enables improved accuracy by confining data transfer strictly within trust boundaries, defined as groups of clients operating under a shared administrative or contractual agreement permitting data sharing.

### III. METHODOLOGY

The proposed HFL framework operates across three hierarchical levels: Clients (IoT wearable devices), Local Aggregators (cluster anchors), and a Global Aggregator. The bottommost layer contains individual devices, each holding a small and statistically heterogeneous local dataset, approximately 30 records in our setting. This combination of limited data volume and cross-client heterogeneity causes standard FL aggregation methods such as FedAvg to produce global models with substantially lower accuracy than centralized training. To address this challenge and reduce the accuracy gap to centralized training, we propose the following framework.

#### A. LSH-Based Cluster Formation

All the members of the consortium will first agree on a suitable machine learning model and architecture. All clients will then individually train the model on their local data for a few warm-up epochs. At this point, each client has a model that has been altered by its own local data. Now, LSH is applied to the model parameters, which will allow it to identify similar edge devices efficiently. Unlike traditional hashing targeting to minimize collisions, LSH aims to ensure that similar types of items are hashed to the same bucket. These similar clients will then be assigned to a single local aggregator (cluster), with the first client designated as the anchor node. This cluster-building process runs once in every global training round. LSH is adopted as the cluster formation mechanism for its sub-linear computational complexity, making it inherently scalable to large-scale IoT deployments where the number of clients may reach thousands. This design choice ensures the framework remains computationally feasible beyond the 33-client experimental cohort evaluated here.

#### B. Within-Cluster Data Movement: Two Configurable Modes for Balancing Privacy and Accuracy

The framework supports two operating modes. In **HFL+NoDataMove (Algorithm 1)**, only model weights are shared between clients and cluster anchors allowing within the cluster model weight aggregation for better privacy. In contrast, in HFL+DataMove, non-anchor clients transfer their local training data to their cluster's anchor node, which then trains on the pooled dataset from all cluster members.

We acknowledge that **HFL+DataMove (Algorithm 2)** relaxes FL's data locality constraint. We do not claim this is pure federated learning. Rather, we present it as one end of a configurable spectrum where practitioners choose the appropriate operating point for their deployment context. Data movement is constrained by design to remain within clusters, which we treat as the trust boundary as reflected in Algorithm 2 (line 9-17). We define a trust boundary as a group of clients that mutually trust each other and consent to sharing raw data within the group. HFL+DataMove is deployed when the

anchor operates within this trust boundary, where clients share data freely as they would within a household, sports team, or workplace wellness programme. HFL+NoDataMove is deployed when the anchor is honest-but-curious, meaning it executes the protocol correctly but may attempt to infer knowledge from received data, placing it outside the trust boundary. By architectural design, everything beyond the cluster boundary, including the global aggregator, is treated as untrusted and receives only aggregated weight vectors. Deployments requiring maximum privacy use NoDataMove; those within established trust domains, such as devices belonging to members of the same household, sports team, or workplace wellness program, can enable data movement for improved accuracy. Additional privacy layers such as differential privacy on transferred data or trusted execution environments at anchors can be introduced in real-life practical implementations.

#### C. Global Aggregation

After cluster-level training (Algorithm 1, lines 9–12 / Algorithm 2, lines 19–22), each anchor transmits its model weights (optionally quantized) to the global aggregator, which aggregates them to produce the updated global model for the round (line 14 / line 24 respectively). The updated global model is shared with all clients to re-initialize local training for the next round. LSH then rebuilds cluster membership based on the updated model weights, allowing clusters to dynamically adapt as models evolve: in HFL+DataMove, data pooling at anchors produces more homogeneous weights causing clusters to consolidate over rounds, whereas in HFL+NoDataMove, weight aggregation alone results in less convergent model parameters leading to more fragmented and variable cluster structures across rounds. This process repeats for  $T$  global rounds until acceptable global model performance is achieved.

The algorithms below present the details of the two proposed approaches: HFL+NoDataMove (Algorithm 1) and HFL+DataMove (Algorithm 2). Figure 1 illustrates their visualization.

#### Algorithm 1: HFL+NoDataMove (Weight Aggregation Only)

**Input:** clients  $K = \{1, \dots, K\}$  with local datasets  $\{D_k\}$ ; rounds  $T$ ; warm-up epochs  $E_w$ ; client-level LSH planes  $P_c$ ; aggregator-level LSH planes  $P_a$

**Output:** final global model  $w^T$

```

1: Initialize global model  $w^0$ 
2: for  $r = 1$  to  $T$  do
3:   for each client  $k$  in parallel do
4:      $w_k^r \leftarrow \text{LocalTrain}(w^{r-1}, D_k, E_w)$ 
5:      $h_k \leftarrow \text{LSHHash}(w_k^r, P_c)$ 
6:   end for
7:    $S^r \leftarrow \text{GroupByHash}(\{h_k\}) \triangleright \text{cluster formation via LSH}$ 
8:    $A^r \leftarrow \text{LSHSelectOrAll}(S^r, P_a, r)$ 
9:   for each cluster  $S_j$  in  $A^r$  do
10:     $n_{S_j} \leftarrow \sum n_k$  for  $k \in S_j$ 
11:     $w_{S_j}^r \leftarrow \sum (n_k / n_{S_j}) w_k^r$  for  $k \in S_j \triangleright \text{weighted average of client models}$ 
12:   end for
13:    $N_{A^r} \leftarrow \sum n_{S_j}$  for  $S_j \in A^r$ 

```

14:  $w^r \leftarrow \Sigma (n_{S_j} / N_A) w_{S_j}^r$  for  $S_j \in A^r \triangleright$  global aggregation

15: end for

16: return  $w^T$

#### Algorithm 2: HFL+DataMove (Within-Cluster Data Pooling)

**Input:** clients  $K = \{1, \dots, K\}$  with local datasets  $\{D_k\}$ ; rounds  $T$ ; warm-up epochs  $E_w$ ; aggregator epochs  $E_a$ ; client-level LSH planes  $P_c$ ; aggregator-level LSH planes  $P_a$

**Output:** final global model  $w^T$

```

1: Initialize global model  $w^0$ 
2: Store original datasets  $B_k \leftarrow D_k$  for all  $k \triangleright$  immutable backup
3: for  $r = 1$  to  $T$  do
4:   for each client  $k$  in parallel do
5:      $w_k^r \leftarrow \text{LocalTrain}(w^{r-1}, D_k, E_w)$ 
6:      $h_k \leftarrow \text{LSHHash}(w_k^r, P_c)$ 
7:   end for
8:    $S^r \leftarrow \text{GroupByHash}(\{h_k\}) \triangleright$  cluster formation via LSH
9:   for all  $k$ , set  $X_k \leftarrow B_k \triangleright$  reset local data buffer
10:  for each cluster  $S_j$  in  $S^r$  do
11:    select anchor  $a_j \in S_j$ 
12:    for each  $m \in S_j \setminus \{a_j\}$  do
13:      payload  $\leftarrow \text{Serialize}(X_m)$ 
14:       $X_{a_j} \leftarrow \text{Concat}(X_{a_j}, \text{Deserialize}(\text{payload})) \triangleright$ 
        anchor accumulates data from each non-anchor member
15:       $X_m \leftarrow \emptyset \triangleright$  non-anchor clears its buffer after transfer
16:    end for
17:  end for
18:   $A^r \leftarrow \text{LSHSelectOrAll}(S^r, P_a, r)$ 
19:  for each cluster  $S_j$  in  $A^r$  do
20:     $n_{S_j} \leftarrow \Sigma n_k$  for  $k \in S_j$ 
21:     $w_{S_j}^r \leftarrow \text{LocalTrain}(w^{r-1}, X_{a_j}, E_a) \triangleright$  anchor trains on pooled data
22:  end for
23:   $N_{A^r} \leftarrow \Sigma n_{S_j}$  for  $S_j \in A^r$ 
24:   $w^r \leftarrow \Sigma (n_{S_j} / N_{A^r}) w_{S_j}^r$  for  $S_j \in A^r \triangleright$  global aggregation
25: end for
26: return  $w^T$ 

```

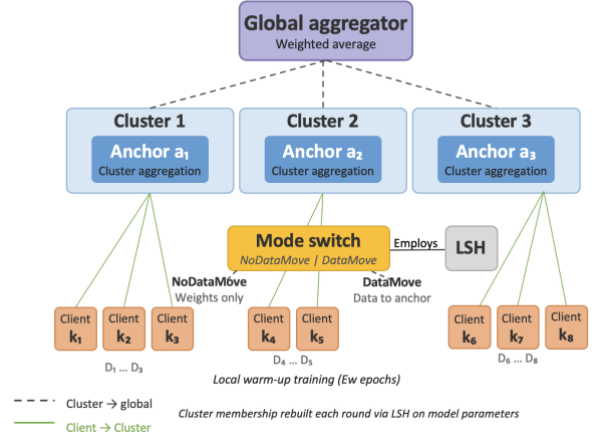


Fig. 1. Proposed Hierarchical Federated Learning architecture. Clients are grouped into clusters via LSH on model parameters after warm-up training. The framework supports two operating modes: HFL+NoDataMove (model weights only) and HFL+DataMove (data transferred to a cluster anchor). Cluster-level models are aggregated at the global server each round.

## IV. EXPERIMENTAL SETUP

### A. Dataset

A Fitbit dataset from Kaggle [22], comprising 33 distinct users and 940 daily records collected over approximately one month, was used to evaluate the proposed framework. As some users had missing daily entries, the number of records per user was not uniform. The dailyActivity\_merged.csv file in the dataset provides features such as TotalSteps, TotalDistance, various activity-specific distance metrics, and duration features (VeryActiveMinutes, FairlyActiveMinutes, LightlyActiveMinutes, SedentaryMinutes). The amount of calories burned per day is used as the regression target. This dataset represents a practical IoT FL scenario, where each user's wearable holds a small, privacy-sensitive local dataset. Although this dataset has mainly been used for centralized wearable-data analytics and prediction tasks, it is well suited for simulating federated learning settings due to its naturally partitioned, user-specific structure.

### B. Baselines and Configuration

All We compare HFL+DataMove against five methods: (1) **Centralized**: all data pooled at a single server (upper bound); (2) **FedAvg**; (3) **FedProx**; (4) **SCAFFOLD**; (5) **HFL+NoDataMove**: our framework with LSH-based cluster formation but weight aggregation only, serving as the direct ablation baseline for data movement.

All methods use the same linear regression model with StandardScaler on features (normalizing each to zero mean and unit variance to prevent larger-magnitude features from dominating) and MinMaxScaler on the target (to bound calorie output to  $[0,1]$  for training stability), ensuring consistent preprocessing across all compared methods. Each client's dataset is split using a temporal 80/20 train/validation strategy, preserving the natural chronological order of observations. We purposely avoid cross-validation, to preserve the natural chronological structure of the data and avoid potential leakage of future observations into training. Training proceeds for 10 global rounds with 10 local warm up epochs. All experiments are conducted across 10 independent random seeds to control randomness at different stages, including model weight initialization and LSH hyperplane generation to ensure statistical robustness of the results. We

report the mean  $\pm$  standard deviation of the final validation MSE. A one-way ANOVA tests for overall group differences, followed by pairwise Welch's t-tests (accommodating unequal variances) with Cohen's d to quantify effect sizes.

## V. EXPERIMENTS AND RESULTS

### A. Benchmarking Proposed Methodology

Table I presents the main results across all six methods. The one-way ANOVA yields  $F = 14.68$  with  $p < 0.001$ , confirming highly significant differences among the methods under discussion.

TABLE I. MEAN VALIDATION MSE RESULTS OVER 10 INDEPENDENT SEEDS

| Method              | Mean MSE      | Std           | Min           | Sig | Cohen's d |
|---------------------|---------------|---------------|---------------|-----|-----------|
| Centralized         | 0.0575        | 0.0012        | 0.0556        | *   | 1.318     |
| <b>HFL+DataMove</b> | <b>0.0701</b> | <b>0.0134</b> | <b>0.0533</b> | —   | —         |
| HFL+NoDataMove      | 0.3125        | 0.1315        | 0.0903        | *** | -2.594    |
| SCAFFOLD            | 0.2490        | 0.0979        | 0.0688        | *** | -2.560    |
| FedAvg              | 0.3132        | 0.1319        | 0.0902        | *** | -2.594    |
| FedProx             | 0.3132        | 0.1319        | 0.0902        | *** | -2.594    |

Sig = significance vs HFL+DataMove (Welch's t-test). \*\*\* =  $p < 0.001$ , \* =  $p < 0.05$ .

Cohen's d is computed as (mean MSE of HFL+DataMove - mean MSE of the comparison method) / pooled standard deviation. Negative values indicate worse MSE than HFL+DataMove, while positive values indicate better MSE.

Table I shows that HFL+DataMove approaches centralized accuracy with a mean MSE of 0.0701 compared to 0.0575 for Centralized. Pairwise tests show large effect sizes (ranging from  $|d| = 2.56$  to 2.59) against FedAvg, FedProx, SCAFFOLD, and HFL+NoDataMove, all significant at  $p < 0.001$ . The comparison with Centralized is significant at  $p = 0.016$  with  $d = 1.318$ , indicating that a mean gap of 0.013 MSE remains. We can observe that FedAvg, FedProx, and HFL+NoDataMove cluster around MSE 0.31 with high variance ( $\text{std} \approx 0.13$ ), reflecting sensitivity to data partitioning. SCAFFOLD performs slightly better (0.2490) due to its variance-reduction mechanism but remains far from centralized accuracy. Also, we can observe that FedAvg and FedProx yield almost identical results (after rounding up to respective decimal points), which is consistent with theoretical expectations: for convex objectives like linear regression, the proximal term in FedProx has no effect on the optimal solution. The best single-seed MSE for HFL+DataMove (0.0533) falls below the Centralized best (0.0556), an interesting observation that is most likely due to random seed variation on a small dataset.

### B. Convergence Analysis

To study the convergence behavior, we tracked the validation MSE across all 10 global rounds. Fig. 2 illustrates the mean convergence patterns across 10 independent seeds, with shaded regions indicating  $\pm$  standard deviation.

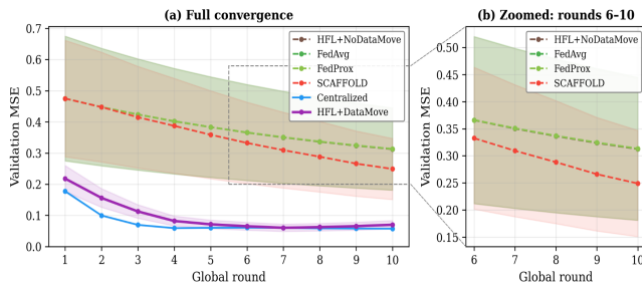


Fig. 2. Mean validation MSE convergence over global rounds (shaded regions:  $\pm$ std across 10 seeds). (a) Full convergence showing all methods. (b)

Zoomed view of rounds 6–10 for separating the closely-tracked FedAvg, FedProx, HFL+NoDataMove, and SCAFFOLD curves.

Examining the convergence patterns more closely, we note that HFL+DataMove converges rapidly in the early rounds, reaching a mean MSE of 0.065 by round 6 and achieving its best mean performance (0.060) at round 7 before exhibiting slight overfitting in later rounds, with the final mean MSE settling at 0.070 by round 10 (Table I). Centralized training follows a similar trajectory with lower variance, stabilizing around 0.058 by round 9. The narrow-shaded region for both HFL+DataMove and Centralized indicates consistent behavior across seed configurations, reinforcing the statistical robustness reported in Table I.

In contrast, the standard FL methods (FedAvg, FedProx, HFL+NoDataMove) exhibit substantially higher mean MSE values throughout, remaining above 0.31 at round 10 with wide confidence bands ( $\text{std} > 0.13$ ). This large variance suggests that these methods are highly sensitive to seed-dependent factors such as client data partitions and model initialization. The zoomed view in Fig. 2(b) reveals that FedAvg, FedProx, and HFL+NoDataMove track each other closely across rounds 6–10, consistent with the near-identical convergence behavior observed between these weight-aggregation-only approaches. SCAFFOLD demonstrates a steeper descent, reaching a mean MSE of 0.249 at round 10, due to its variance-reduction mechanism through control variates. However, even SCAFFOLD remains substantially above both HFL+DataMove and Centralized, confirming that within-cluster data consolidation is the key factor for accuracy improvement in our framework.

The cluster formation dynamics provide further insight into why HFL+DataMove achieves superior accuracy. As shown in Fig. 3, HFL+DataMove consolidates all 33 clients into a single cluster during rounds 3–5. In this phase, the framework effectively approximates centralized training, since pooling all client data at a single anchor is functionally equivalent to centralized aggregation. From round 6 onward, small differences in anchor-level model parameters cause LSH to split the clients into two clusters. This does not indicate a loss of the benefits of consolidation; rather, it reflects LSH adapting to subtle divergences in anchor-trained models. Pooled training still occurs within each cluster, preserving the accuracy benefits of data consolidation while operating over two homogeneous subsets instead of one. In contrast, HFL+NoDataMove generally maintains a 2-cluster structure, with occasional fragmentation into up to 4 clusters during rounds 7–9. This aligns well with HFL+NoDataMove's slight MSE increase after round 8 observed in Fig. 2. This contrast directly explains the accuracy gap: data movement creates a positive feedback loop where pooling leads to more similar models, which leads to larger clusters, which leads to more pooling. HFL+NoDataMove lacks this mechanism since weight aggregation alone cannot achieve the same degree of model convergence. We note that the specific cluster assignments shown here correspond to a single representative seed and are expected to vary across runs due to stochastic LSH hyperplane generation; however, the statistical significance of final MSE gains across 10 seeds ( $p < 0.001$ ) confirms that the accuracy gains are robust to these variations.

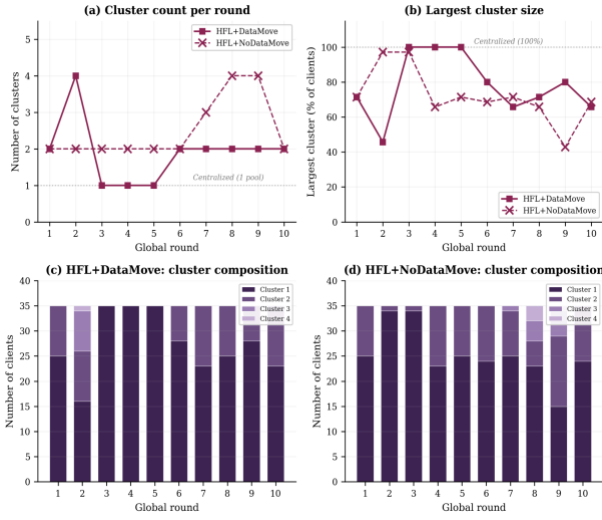


Fig. 3. Cluster formation dynamics across global rounds. (a) Number of clusters per round. (b) Largest cluster as percentage of total clients. (c)–(d) Client distribution across clusters for HFL+DataMove and HFL+NoDataMove.

### C. Communication Cost and Training Time

We quantify communication overhead by measuring the total bytes transferred and the projected transfer time across 10 global rounds. Each payload (model weights or data) is serialized via Python’s pickle protocol, the byte count is recorded, and the projected transfer time is computed as  $(\text{bytes} \times 8) / (\text{bandwidth} \times 10^6) + \text{RTT}$  with a fixed round-trip time of 2 ms per transfer. We note that this uniform model is a simplification: in practice, the Centralized baseline would involve a single bulk data upload rather than per-round exchanges, making its communication pattern structurally different from the federated methods. We apply the same formula to all methods for consistent comparison. A fixed 2 ms RTT-based delay per transfer was assumed to emulate a low-latency edge-network setting, consistent with prior work targeting end-to-end delays on the order of 1 ms in 5G-enabled edge systems [23]. Serialization and deserialization costs were excluded from the reported communication-time totals, as they were found to be insignificant relative to the projected network transfer time. Computation time is measured separately as the cumulative wall-clock time spent on local model training at client and aggregator nodes over all 10 rounds. This enables communication and training costs to be analyzed independently.

Results under 100Mbps emulated link are summarized in Table II and the accuracy-communication trade-off is depicted in Fig. 4.

TABLE II. COMMUNICATION AND TIME OVERHEAD (10 ROUNDS)

| Method              | Bytes          | Comm (s)    | Comp (s)    | Total (s)    |
|---------------------|----------------|-------------|-------------|--------------|
| Centralized         | 24,202         | 0.07        | 7.95        | 8.03         |
| <b>HFL+DataMove</b> | <b>378,912</b> | <b>2.17</b> | <b>8.52</b> | <b>10.68</b> |
| HFL+NoDataMove      | 152,250        | 1.46        | 10.73       | 12.20        |
| FedAvg              | 147,000        | 1.41        | 9.59        | 11.00        |
| FedProx             | 147,000        | 1.41        | 12.01       | 13.42        |
| SCAFFOLD            | 227,040        | 1.34        | 14.42       | 15.76        |

Communication time at 100 Mbps emulated link. Computation in sequential simulation.

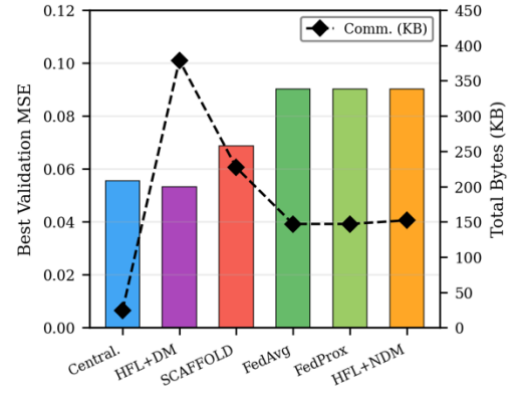


Fig. 4. Accuracy-communication trade-off. Bars: best single-seed validation MSE (lower is better), illustrating each method’s achievable lower bound. Dashed line: total bytes transferred. Table I reports mean MSE for statistical robustness.

HFL+DataMove transfers  $2.6\times$  the bytes compared to FedAvg (379 KB vs 147 KB across 10 rounds), but the absolute cost remains moderate, under 400 KB total. At 100 Mbps, this adds only 0.76 seconds of communication overhead compared to FedAvg. Even at 1 Mbps, a typical IoT edge scenario, the total time for HFL+DataMove is 13.69 seconds, only 12% more than FedAvg’s 12.17 seconds and already faster than FedProx (14.58s) and SCAFFOLD (17.56s) even at this most constrained bandwidth, while delivering a substantial MSE improvement. This represents an acceptable overhead for IoT deployments given the accuracy gain.

It is interesting to observe that HFL+DataMove’s computation time (8.52s) is actually lower than all other FL methods. Data pooling at the anchor enables efficient training: instead of 33 separate local training passes, HFL+DataMove performs only 1–2 aggregator training runs on pooled data per round. With the current compact linear model ( $\sim 700$  bytes), data transfer dominates communication costs. For larger models containing millions of parameters, however, the communication trade-off may change: FedAvg’s communication cost scales with model size, while HFL+DataMove’s scales with dataset size rather than model size, making it potentially more efficient for large models trained on compact datasets. Further investigation with larger models is needed to confirm this behavior.

### D. Bandwidth Sensitivity Analysis

Since HFL+DataMove transfers raw data within clusters in addition to model weights, it is important to understand whether this additional communication overhead becomes a bottleneck under constrained network conditions which is typical for IoT deployments. To evaluate this, we use a unified communication cost emulation framework applied consistently across all six methods.

We then project the total training time (computation + communication) across six bandwidth tiers from 1 Mbps to 1 Gbps by recomputing only the communication component at each tier while keeping the measured computation time fixed. Table III presents these results.

TABLE III. TOTAL TRAINING TIME (S) ACROSS BANDWIDTH TIERS (10 ROUNDS)

| Method               | 1 Mbps       | 10 Mbps      | 50 Mbps      | 100 Mbps     | 500 Mbps     | 1 Gbps       |
|----------------------|--------------|--------------|--------------|--------------|--------------|--------------|
| Centralized          | 8.22         | 8.04         | 8.03         | 8.03         | 8.02         | 8.02         |
| <b>HFL+Data Move</b> | <b>13.69</b> | <b>10.96</b> | <b>10.72</b> | <b>10.68</b> | <b>10.66</b> | <b>10.66</b> |
| HFL+NoDataMove       | 13.40        | 12.31        | 12.21        | 12.20        | 12.19        | 12.18        |
| FedAvg               | 12.17        | 11.11        | 11.02        | 11.00        | 10.99        | 10.99        |
| FedProx              | 14.58        | 13.52        | 13.43        | 13.42        | 13.41        | 13.41        |
| SCAFFOLD             | 17.56        | 15.92        | 15.78        | 15.76        | 15.74        | 15.74        |

Notably even at the most constrained bandwidth of 1 Mbps, which represents a realistic worst case for IoT edge networks, HFL+DataMove completes training in 13.69 s — comparable to HFL+NoDataMove (13.40 s) and only 12% more than FedAvg (12.17 s). At bandwidths of 10 Mbps and above, HFL+DataMove (10.96 s) is actually faster overall than HFL+NoDataMove (12.31 s), FedProx (13.52 s), and SCAFFOLD (15.92 s), as its lower computation time offsets the slightly higher data transfer cost. SCAFFOLD incurs the highest total time at every bandwidth tier due to its control variate exchanges which transfers approximately 1.5× the bytes of FedAvg per round. Beyond 100 Mbps, all methods converge confirming that computation dominates over communication for this model and dataset scale. These results indicate that the total of 378 KB transferred by HFL+DataMove across all 10 global rounds does not represent a practical bottleneck at any reasonable deployment bandwidth, further supporting the feasibility of the proposed approach for IoT environments.

Based on these results, practitioners can follow the following deployment heuristic: HFL+NoDataMove is recommended when data governance policies prohibit raw data leaving client devices, and HFL+DataMove when operating within an established trust boundary with network bandwidth of at least 1 Mbps, where it incurs only 12% total time overhead relative to FedAvg while delivering substantially superior accuracy (mean MSE 0.0701 vs 0.3132).

## VI. DISCUSSION

Experimental results indicate that within-cluster data pooling (HFL+DataMove), guided by LSH-based client grouping, enables the proposed framework to approach centralized accuracy while maintaining a distributed architecture. Cluster formation dynamics reveal a clear pattern in HFL+DataMove: data pooling at anchors produces more homogeneous model parameters, which causes LSH to group clients into fewer, larger clusters, which in turn leads to even more comprehensive pooling which effectively approximating centralized training. This is evidenced by all 33 clients merging into a single cluster during rounds 3–5 before stabilizing into two clusters, particularly expected for linear models where pooled training converges toward the centralized optimum due to the convex optimization landscape.

The results suggest that the primary bottleneck in standard FL is not the aggregation algorithm itself but the heterogeneity of data each client trains on, pooling within trust boundaries directly addresses this. The communication overhead introduced by data movement remains negligible relative to

the accuracy gain, representing an acceptable trade-off for IoT deployments. Furthermore, the adaptive behavior of LSH-based cluster formation demonstrates that the framework self-organizes without requiring a fixed number of clusters, consolidating when client models converge and fragmenting when they diverge.

One of the fundamental questions that may raise concerns on proposed HFL+DataMove methodology will be the compatibility of data movement with FL’s core principles. We do not claim this is pure FL, rather we relax the data locality constraint (raw data not leaving the origin) in the original FL design [1]. We emphasize this constraint was a design choice for a specific threat model (mobile keyboard prediction), not an explicit rule the FL should adhere [24]. HFL+DataMove is better characterized as controlled partial centralization, where raw data movement is strictly bounded within pre-agreed trust boundaries rather than flowing to a central server. Further, gradient inversion attacks [21] have shown that shared model weights can reconstruct training examples, reducing the privacy distinction between sharing weights and sharing raw data. Our proposed framework makes this choice explicit and configurable to the practitioners depending on the priority (privacy vs accuracy) with the choice between NoDataMove (standard FL privacy) and DataMove (improved accuracy within trust boundaries), respectively. Further improvement to DataMove on the privacy axis is possible by incorporating additional privacy layers like differential privacy on transferred data, trusted execution environments at anchors, or secure multi-party computation within clusters [25]. Additionally, a policy layer [26] could also be integrated to DataMove algorithm to have a systematic control over definition of trust boundaries, enhancing practical deployability. This configurability aligns with the Industry 5.0 vision of human-centric technology where users and operators retain control over deployment decisions.

Several limitations should also be noted. The experiments use a linear regression model on a relatively small tabular dataset. For linear models, data pooling is expected to approximate centralized training closely due to the convex optimization landscape. We expect the accuracy gap between HFL+DataMove and flat baselines to narrow but persist for deep models; pooled training no longer automatically reaches the centralized optimum, though anchors still benefit from larger, more homogeneous training sets within each cluster, which reduce local variance and produce more reliable model updates to the global aggregator. Further experiments should be conducted to validate the transferability of these strong results to more complex deep neural networks where non-convexity, local minima, and more complex data heterogeneity dynamics may appear.

## VII. CONCLUSION AND FUTURE WORK

This work has presented a Hierarchical Federated Learning framework with a configurable privacy-accuracy spectrum which allows within-cluster model weight or data pooling guided by LSH-based client grouping, to prioritize one of the above two aspects based on the requirement of the end user. The two alternatives of the algorithms, along with centralized and three standard FL benchmarks were evaluated on a Fitbit wearable activity dataset representing a practical IoT application. Repeated experiments for 10 different seeds and statistical analysis for significance are in favor of the proposed HFL+DataMove method that approaches centralized accuracy (mean MSE 0.070 vs 0.057) while

significantly outperforming all FL baselines with large effect sizes. This variant further converges rapidly, reaching a best mean MSE of 0.060 by round 7 reaching close performance to centralized variant while FedAvg, FedProx, and HFL+NoDataMove remain around MSE 0.31, and even SCAFFOLD reaches only 0.249. With respect to communication cost and training time, HFL+DataMove transfers only  $2.6\times$  the bytes compared to FedAvg (379 KB vs 147 KB across 10 global rounds) and the overhead remains under 400 KB. Further, this algorithm remains practical across a wide range of network bandwidths which is evident compared to FedAvg where HFL+DataMove demands only 13.69s (only 12% more) for complete training. Interestingly, at 10 Mbps and above, HFL+DataMove is faster overall than FedProx, SCAFFOLD, and HFL+NoDataMove due to its lower computation time in consolidated anchor training, and comparable to FedAvg.

Several directions for future work emerge from this study. Evaluating the framework across models of increasing complexity, from regularized regression variants and MLPs to deeper architectures such as CNNs and transformers, as well as on non-tabular data modalities, would extend generalizability beyond the linear models employed here. Policy-aware cluster formation with trust matrices could extend the approach to multi-institutional deployments. Integrating differential privacy on transferred data would strengthen the privacy guarantees of the DataMove variant and would be empirically evaluated to quantify the accuracy-privacy trade-off under formal  $\epsilon$ -differential privacy guarantees. The overfitting observed after round 7 suggests that early stopping mechanisms could further improve accuracy. Developing client dropout resilience would be essential for production IoT deployments where device availability is intermittent. Finally, exploring heterogeneity-aware techniques such as StatAvg [27] at the cluster level, which normalises feature distributions across clients within each LSH-formed cluster, could further improve model convergence under within-cluster data heterogeneity.

## REFERENCES

- [1] M. Breque, L. De Nul, and A. Petridis, "Industry 5.0: Towards a sustainable, human-centric and resilient European industry," European Commission, Directorate-General for Research and Innovation, Luxembourg, 2021, doi: 10.2777/308407.
- [2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in Proc. AISTATS, 2017, pp. 1273–1282.
- [3] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in Proc. MLSys, 2020.
- [4] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," in Proc. ICML, 2020, pp. 5132–5143.
- [5] B. Soltani, V. Haghighi, A. Mahmood, Q. Z. Sheng, and L. Yao, "A survey on participant selection for federated learning in mobile networks," in Proc. 17th ACM Workshop on Mobility in the Evolving Internet Architecture, Oct. 2022, pp. 19–24.
- [6] Z. Jiang, P. Ye, S. He, W. Wang, R. Chen, and B. Li, "Lotto: Secure participant selection against adversarial servers in federated learning," arXiv:2401.02880, 2024.
- [7] S. M. Shah and V. K. Lau, "Model compression for communication efficient federated learning," IEEE Trans. Neural Netw. Learn. Syst., 2021.
- [8] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," arXiv:1610.05492, 2016.
- [9] Z. Qiao, X. Yu, J. Zhang, and K. B. Letaief, "Communication-efficient federated learning with dual-side low-rank compression," arXiv:2104.12416, 2021.
- [10] S. Ji, W. Jiang, A. Walid, and X. Li, "Dynamic sampling and selective masking for communication-efficient federated learning," IEEE Intell. Syst., vol. 37, no. 2, pp. 27–34, 2021.
- [11] D.-D. Le et al., "A survey of model compression and its feedback mechanism in federated learning," in Proc. 5th ACM Workshop on Intelligent Cross-Data Analysis and Retrieval (ICDAR '24), Phuket, Thailand, Jun. 2024, pp. 1–6, doi: 10.1145/3643488.3660293.
- [12] R. Chen, Q. Wan, P. Prakash, L. Zhang, X. Yuan, Y. Gong, X. Fu, and M. Pan, "Workie-Talkie: Accelerating federated learning by overlapping computing and communications via contrastive regularization," in Proc. IEEE/CVF ICCV, 2023, pp. 16999–17009.
- [13] J. Jia, J. Liu, C. Zhou, H. Tian, M. Dong, and D. Dou, "Efficient asynchronous federated learning with sparsification and quantization," Concurrency Comput.: Pract. Exp., p. e8002, 2023.
- [14] J. Liu, Q. Zeng, H. Xu, Y. Xu, Z. Wang, and H. Huang, "Adaptive block-wise regularization and knowledge distillation for enhancing federated learning," IEEE/ACM Trans. Netw., 2023.
- [15] I. Kaur and A. J. Jadhav, "Federated learning in IoT: A survey from a resource-constrained perspective," arXiv:2308.13157, Aug. 2023.
- [16] J. Zhang, J. Wang, Y. Li, F. Xin, F. Dong, J. Luo, and Z. Wu, "Addressing heterogeneity in federated learning with client selection via submodular optimization," ACM Trans. Sensor Netw., vol. 20, no. 2, pp. 1–32, 2024.
- [17] Z. Zhong, W. Bao, J. Wang, X. Zhu, and X. Zhang, "Flee: A hierarchical federated learning framework for distributed deep neural network over cloud, edge, and end device," ACM Trans. Intell. Syst. Technol., vol. 13, no. 5, pp. 1–24, 2022.
- [18] R. Saha, S. Misra, and P. K. Deb, "FogFL: Fog-assisted federated learning for resource-constrained IoT devices," IEEE Internet Things J., vol. 8, no. 10, pp. 8456–8463, 2020.
- [19] A. Kumar and S. N. Srirama, "Fog enabled distributed training architecture for federated learning," in Big Data Analytics: 9th Int. Conf., BDA 2021, Proc. 9, Springer, 2021, pp. 78–92.
- [20] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," IEEE Commun. Surveys Tuts., vol. 22, no. 3, pp. 2031–2063, 2020.
- [21] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in Proc. NeurIPS, 2019, pp. 14774–14784.
- [22] M. Jacob, "FitBit Fitness Tracker Data," Kaggle, 2016. [Online]. Available: <https://www.kaggle.com/datasets/arashnic/fitbit>
- [23] A. Parvez, A. Rahmati, I. Güvenç, A. I. Sarwat, and H. Dai, "A survey on low latency towards 5G: RAN, core network and caching solutions," IEEE Commun. Surveys Tuts., vol. 20, no. 4, pp. 3098–3130, 2018, doi: 10.1109/COMST.2018.2841349.
- [24] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," arXiv:1811.03604, 2019.
- [25] X. Ren, S. Yang, C. Zhao, J. McCann, and Z. Xu, "Belt and braces: When federated learning meets differential privacy," Commun. ACM, vol. 67, no. 12, pp. 66–77, Nov. 2024, doi: 10.1145/3650028.
- [26] P. He and M. Capretz, "DPFedBank: Crafting a privacy-preserving federated learning framework for financial institutions with policy pillars," arXiv:2410.13753, Oct. 2024.
- [27] P. S. Bouzinis, P. Radoglou-Grammatikis, I. Makris, T. Lagkas, V. Argyriou, G. T. Papadopoulos, P. Sarigiannidis, and G. K. Karagiannidis, "StatAvg: Mitigating data heterogeneity in federated learning for intrusion detection systems," IEEE Trans. Netw. Serv. Manag., Apr. 2025, doi: 10.1109/TNSM.2025.356438.