Computer Modeling in
Engineering & Sciences

Tech Science Press

Doi:10.32604/cmes.2025.065833

ARTICLE

A Novel Clustered Distributed Federated Learning Architecture for Tactile Internet of Things Applications in 6G Environment

Omar Alnajar* **and Ahmed Barnawi**

Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, 80221, Saudi Arabia

*Corresponding Author: Omar Alnajar. Email: oorabialnajar@stu.kau.edu.sa

Received: 22 March 2025; Accepted: 09 June 2025; Published: 30 June 2025

ABSTRACT: The Tactile Internet of Things (TIoT) promises transformative applications—ranging from remote surgery to industrial robotics—by incorporating haptic feedback into traditional IoT systems. Yet TIoT's stringent requirements for ultra-low latency, high reliability, and robust privacy present significant challenges. Conventional centralized Federated Learning (FL) architectures struggle with latency and privacy constraints, while fully distributed FL (DFL) faces scalability and non-IID data issues as client populations expand and datasets become increasingly heterogeneous. To address these limitations, we propose a Clustered Distributed Federated Learning (CDFL) architecture tailored for a 6G-enabled TIoT environment. Clients are grouped into clusters based on data similarity and/or geographical proximity, enabling local intra-cluster aggregation before inter-cluster model sharing. This hierarchical, peer-to-peer approach reduces communication overhead, mitigates non-IID effects, and eliminates single points of failure. By offloading aggregation to the network edge and leveraging dynamic clustering, CDFL enhances both computational and communication efficiency. Extensive analysis and simulation demonstrate that CDFL outperforms both centralized FL and DFL as the number of clients grows. Specifically, CDFL demonstrates up to a 30% reduction in training time under highly heterogeneous data distributions, indicating faster convergence. It also reduces communication overhead by approximately 40% compared to DFL. These improvements and enhanced network performance metrics highlight CDFL's effectiveness for practical TIoT deployments. These results validate CDFL as a scalable, privacy-preserving solution for next-generation TIoT applications.

KEYWORDS: Distributed federated learning; Tactile Internet of Things; clustering; peer-to-peer

1 Introduction

The Tactile Internet of Things (TIoT) integrates haptic and kinesthetic feedback into traditional IoT systems—long confined to text, audio, and video exchanges—to enable real-time, touch-sensitive interactions across healthcare [1], robotics, and entertainment domains [2,3]. Unlike conventional IoT's low-power sensor networks, TIoT connects sensors, actuators, and robots to deliver simultaneous tactile and visual feedback, demanding ultra-low latency, high reliability, robust security, and continuous availability [4,5]. Predicting tactile signals is crucial: by forecasting contact events or force changes, intelligent agents can preemptively adjust, thereby reducing delay and enhancing reliability in teleoperation, robotic manipulation, and remote surgery [6]. These stringent requirements and dynamic conditions drive the need for novel architectures and methods [7,8].

Fifth-generation (5G) networks, which offer up to 20 Gbps and 8 ms latency under ideal conditions, fall short of the terabit-scale bandwidth and submillisecond latency needs of TIoT for applications such as holographic communication and autonomous vehicle swarms [9,10]. Limitations in spectral efficiency, device density support, and energy use further expose 5G's inadequacy for ultra-reliable low-latency communication (URLLC).

To bridge this gap, researchers explore enhancements and new paradigms: Federated Learning (FL) [11], network slicing [12,13], virtualized network functions [14,15], and software-defined networking [16]. In the transition from 5G to sixth-generation (6G), cloud-native designs with dynamic resource allocation and edge computing will enable Clustered Distributed Federated Learning (CDFL) for TIoT, satisfying URLLC demands via adaptive client clustering and localized processing [17]. Advanced 6G slicing techniques create tailored virtual networks for TIoT without degrading overall performance [18].

We propose a CDFL framework for TIoT that reduces the time to convergence, mitigate the Non-IId and scalability challenges, and saves the cost. Thus, it will realize future green and sustainable production factories [19]. While FL has proven effective in smart healthcare, transportation, Unmanned Aerial Vehicle (UAV), and industrial IoT [20], TIoT's unique latency and heterogeneity challenges require DFL and clustering to reduce overhead, enhance privacy, and improve convergence under non-IID data [21–24].

Building on our prior in-content aware FL design for 6G [25], this study delivers the first evaluation of DFL and CDFL in TIoT settings. We model their computational and communication costs, demonstrating that while DFL matches centralized accuracy at higher overhead, CDFL mitigates degradation as scale and data heterogeneity grow, offering compensatory mechanisms for system underperformance.

The main contributions of this work are:

1. Developing a fully distributed peer-to-peer architecture and workflow for TIoT applications.
2. Comparing the proposed DFL framework with traditional centralized FL architectures across various performance metrics.
3. Proposing a CDFL architecture for IID scenarios to enhance scalability and streamline communication and aggregation processes.
4. Addressing non-independent and identically distributed (non-IID) data challenges by grouping clients into homogeneous clusters based on dataset similarities.
5. Analyzing the computational and communication cost factors associated with both DFL and CDFL solutions.

This paper is organized as follows. Section 2 reviews the related work. Section 3 presents the proposed fully distributed and clustered FL architectures along with their operational workflows. Section 4 details the methodology, including the use case, dataset generation, and experimental settings. Section 5 provides a comprehensive evaluation of the proposed approaches, analyzing their performance and complexity. Finally, Section 6 concludes the paper and outlines potential future research directions. Fig. 1 detailed this structure.

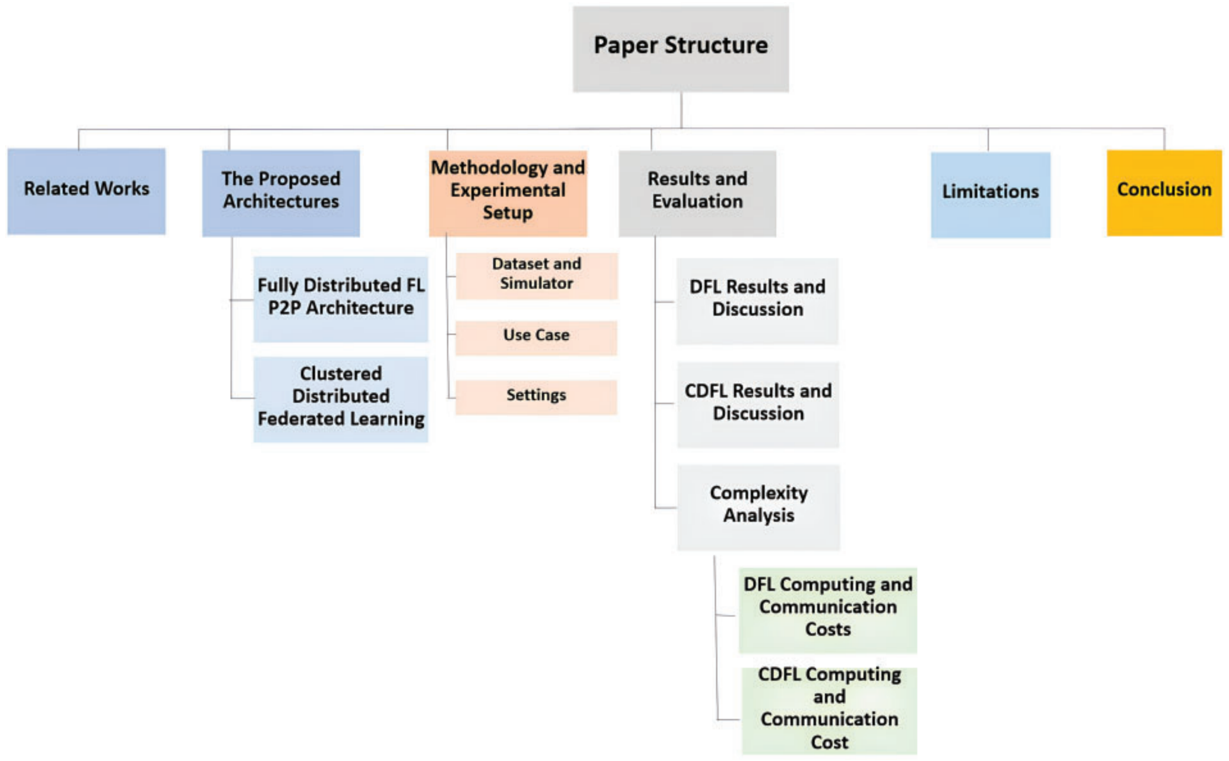
Paper Structure
Related Works
The Proposed Architectures
Fully Distributed FL P2P Architecture
Clustered Distributed Federated Learning
Methodology and Experimental Setup
Dataset and Simulator
Use Case
Settings
Results and Evaluation
DFL Results and Discussion
CDFL Results and Discussion
Complexity Analysis
DFL Computing and Communication Costs
CDFL Computing and Communication Cost
Limitations
Conclusion

Figure 1: Paper structure

2 Related Works

Several recent works in the literature have addressed the communication mechanisms of DFL. One such mechanism is peer-to-peer (P2P) communication—where participants interact directly without a central authority. For instance, reference [26] extended the primary centralized FL algorithm, federated averaging (FedAvg), to demonstrate the potential of P2P communication by presenting Decentralized Federated Trusted Averaging (DeFTA), a decentralized FL framework that serves as a plug-and-play replacement for FedAvg. Behera et al. [27] leveraged a consensus algorithm called RAFT for aggregator selection to add a secure layer for model delivery within the network. However, this approach inevitably increases communication costs, which reference [28] sought to reduce. Similarly, to enhance communication efficiency, Gupta et al. [29] employed a graph theoretical framework.

All the aforementioned works assume homogeneous devices, whereas realistic environments inherently exhibit heterogeneity. To address this challenge, other techniques have been proposed. For example, reference [30] tackled non-IID scenarios by partitioning them into multiple IID clusters and performing aggregation within each cluster. Meanwhile, Liu et al. [31] combined adaptive clustering with a hierarchical DFL structure for Internet of Vehicles (IoV) applications. On a similar note, Wu et al. [32] incorporated deep reinforcement learning-based adaptive staleness control along with a heterogeneity-aware client-edge association strategy to improve system efficiency and mitigate the adverse effects of staleness without compromising model accuracy. Furthermore, Chen et al. [33] addressed slow convergence and poor learning performance by proposing a framework that enhances learning fairness and system efficiency via model exchanges among local clients and adaptive aggregation based on performance.

Clustering is one of the most common strategies to mitigate the impact of heterogeneity, offering advantages in both scalability and communication efficiency. The principal objective of our work is to develop an efficient CDFL architecture suitable for large-scale and non-IID scenarios.

Recent literature makes significant contributions to this field, aligning with our goal of developing efficient solutions for non-IID scenarios. For example, Solat et al. [34] focused on improving the convergence speed and efficiency of mobile traffic prediction models while minimizing the impact of stragglers. Wang et al. [35] introduced a Clustered Federated Learning (CFL) paradigm combined with model ensemble techniques within the Open Radio Access Network (O-RAN) architecture to enhance the generalization performance of FL models. In addition, Morafah et al. [36] exploited inference similarity among client models to form clusters based on the similarity of their learning tasks, while Zhao et al. [37] proposed an ensemble FL paradigm to reduce the divergence caused by non-IID data by forming individual clusters without explicitly grouping them based on inference similarity.

To address scalability issues, Chen et al. [38] employed hierarchical aggregation to improve communication efficiency and introduced mechanisms such as authenticated encryption, a random pairwise key scheme, and key revocation to enhance security. In the realm of network anomaly detection in large-scale networks, Sáez-de-Cámara et al. (2023) [18] leveraged unsupervised device clustering combined with autoencoder neural networks.

The trade-off between performance (in terms of model accuracy) and communication overhead has been the primary focus of several studies [39–41]. For example, reference [39] utilized hierarchical aggregation to balance these factors, whereas Ouyang et al. [40] adopted cluster-wise straggler dropout and correlation-based node selection. Additionally, the concept of dynamic clustering, which adapts to real-time environmental changes and determines the optimal cluster partitioning without pre-specifying the number of clusters, was explored in [42].

In all cases, due to the distributed nature of FL processing, both computing and communication overheads must be meticulously considered. Although some of the aforementioned works provide analyses of one or both of these complexity factors, a more in-depth evaluation is warranted—and is provided herein. Furthermore, to the best of our knowledge, existing works still rely on a central server as the final aggregator in the FL process, an approach that not only increases communication costs but also compromises reliability. Our work offloads the entire aggregation process to the network edge (TIoT clients) in a fully P2P manner, leveraging their capabilities and reducing network burdens. In addition, we integrate this solution with the envisioned 6G enablers, thereby underscoring the advantages of our proposed framework.

Table 1 presents a comparative analysis of our proposed CDFL architecture against state-of-the-art approaches cited in the literature. The table highlights key metrics and innovations, emphasizing the unique advantages of the CDFL framework in terms of scalability, non-IID data handling, and communication efficiency within 6G-enabled TIoT environments.

Table 1: Comparative analysis of CDFL against state-of-the-art FL architectures

Approach	**Key features**	**Scalability**	**Non-IID handling**	**Comm. overhead**	**Central server**	**Ref.**
			Existing Approaches			
Non-IID clustering	Breaks non-IID data into IID clusters	Moderate	Partial	High	Yes	[30]

(Continued)

Table 1 (continued)

Approach	Key features	Scalability	Non-IID handling	Comm. overhead	Central server	Ref.
Hierarchical DFL for IoV	Adaptive clustering with edge hierarchy	High	Partial	Moderate	Yes	[31]
Inference similarity clustering	Clusters clients via model similarity	Low	Yes	High	Yes	[36]
Ensemble FL	Reduces divergence via ensemble clusters	Moderate	Yes	Moderate	Yes	[37]
Hierarchical aggregation	Secure hierarchical aggregation	High	No	Low	Yes	[38]
Cluster-wise straggler dropout	Straggler mitigation via correlation	Moderate	Partial	Moderate	Yes	[40]
Dynamic adaptive clustering	Dynamic cluster partitioning	High	Partial	Moderate	Yes	[42]
Proposed CDFL Architecture						
CDFL	• Clustering based on data similarity/geography • Fully decentralized P2P aggregation • 6G-enabled edge computing and slicing • Intra/inter-cluster hierarchical aggregation • Comprehensive cost analysis	**High**	**Yes**	**Low**	**No**	**This Work**

3 The Proposed Architectures

This section provides a detailed examination of the lower layer, emphasizing the capability for clients to independently perform learning and inference tasks prior to engaging the virtual Edge and Core components. Additionally, a portion of the aggregation process is offloaded to the client side, thereby conserving valuable resources at the edge. Accordingly, our approaches focus on cross-device FL to maintain data privacy and reduce communication overhead as well as computational resources on the access network—an aspect that is critical for TIoT applications. In this context, we outline two principal approaches: one for a fully distributed P2P FL solution and another that enhances DFL through CDFL.

It is important to note that the term *Fully Distributed* emphasizes that the raw data remain stored at their sources, namely the tactile devices, rather than on any Edge or Core network entities. Although the clustering mechanisms and related coordination tasks for aggregation may leverage select Virtual Network Functions (VNFs), the primary focus remains on ensuring that data storage is localized.

3.1 Fully Distributed FL P2P Architecture

This section details the proposed architecture for fully distributed FL using the P2P method. Initially, we review the reference centralized FL architecture, which serves as the baseline for comparison.

Fig. 2 illustrates the foundational centralized FL architecture where each client trains its local model, then upload it to the central server to be aggregated. Once the aggregation is completed, the global model will be downloaded to each client. The Algorithm 1 outlines its primary steps which will be used in the experiment for implementing the centralized method as a baseline architecture. The process commences with the initialization of the global model weights w_{g_0} and proceeds iteratively over multiple rounds. In each round, a predetermined number of clients m is randomly selected from the available pool. Each selected client k then updates its local model weights in parallel based on the current global model w_g, as depicted in Algorithm 2. Subsequently, the updates from the clients are aggregated to form the new global model weights using the following equation:

$$w_g = \sum_{k \in S_t} \left(\frac{n_k}{n_t} \right) \cdot w_l^{(k)} \tag{1}$$

where n_k denotes the number of data points held by client k and n_t represents the total number of data points across the selected clients. This weighted aggregation ensures that each client's contribution is proportionately reflected in the updated global model.

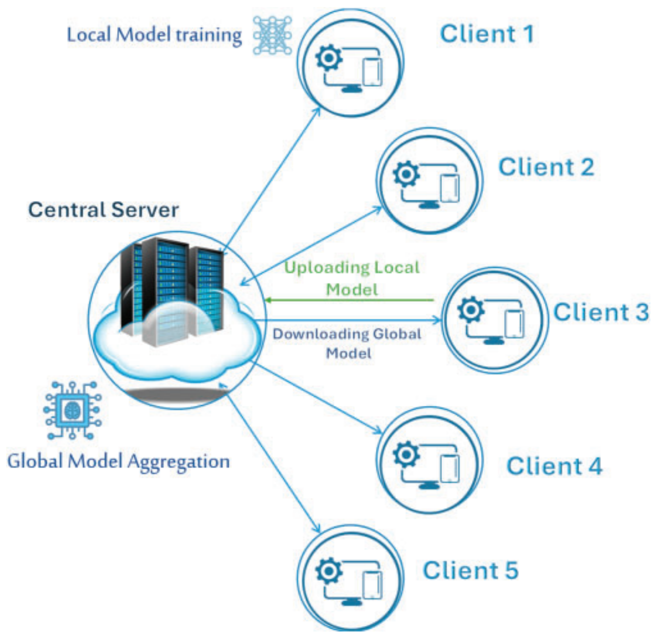

Figure 2: Centralized federated learning

Algorithm 1: Centralized federated learning

```
  Output: Updated global model weights $w_g$
1 Initialize global model weights $w_g \leftarrow w_{g_0}$;
2 for each round $t = 1, 2, \ldots$ do
```

(Continued)

Algorithm 1 (continued)

```
3       Determine the number of clients to participate in this round, $m$;
4       Select a random set of $m$ clients, $S_t$;
5       **foreach** *client k in parallel from* $S_t$ **do**
6           Download the global model $w_g$ from the server;
7           Update client model weights: $w_l \leftarrow \text{LocalModelUpdate}(k, w_g)$;
8           Upload the updated local model to the central server;
9       **end**
10      Aggregate client updates to update global model weights:
        $w_g \leftarrow \sum_{k \in S_t} \left(\frac{n_k}{n_t}\right) \cdot w_l^{(k)}$
11 **end**
12 **return** $w_g$
```

Algorithm 2: Local model update function

```
   **Input:** Client index $k$, Initialize global model weights $w_g \leftarrow w_{g_0}$
1 Split client $k$'s data into batches of size $B$
2 **for** *each local epoch i from* 1 to $E$ **do**
3       **for** *each batch b in batches* **do**
4           Update local model weights: $w_l \leftarrow \text{average}(w_l, w_{g_0})$;
5       **end**
6  **end**
7  **return** *Updated local model weights* $w_l$
```

Fig. 3 presents the fully decentralized P2P FL architecture where each client shares its model after local training to other peers to be aggregated locally and build the global model. In this P2P framework, Algorithm 3 illustrates the distributed learning process executed across multiple client devices. Each client independently initializes its local model and sets the required hyper-parameters, then trains its model on the locally stored data by performing forward and backward passes to update its weights. Instead of transmitting model parameters to a central server, clients share their parameters with randomly selected peers and perform averaging to update the model weights. After each round of local training and aggregation, clients validate their models on local validation sets. This iterative cycle of local training, P2P communication, and aggregation continues over several global epochs. A convergence check is performed at each round; training is halted once the average accuracy across clients surpasses a predefined threshold, otherwise, the process iterates until convergence is achieved.

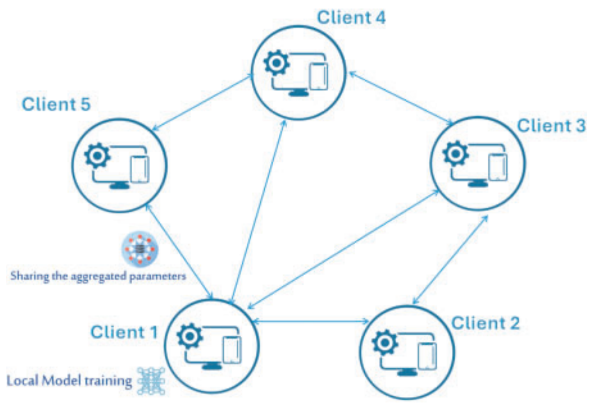

Figure 3: Fully distributed federated learning

Algorithm 3: P2P fully decentralized federated learning

```
   Output: Updated global model weights $w_g$
1  Initialize global model weights $w_g \leftarrow w_{g_0}$;
2  for each round $t = 1, 2, \ldots$ do
3      Determine randomly selected clients $S_t$ for this round;
4      $W \leftarrow [], N \leftarrow []$; // Local weights and data sizes
5      foreach client k in parallel from $S_t$ do
6          $w_l^{(k)} \leftarrow$ Copy of global weights $w_g$;
7          Train $w_l^{(k)}$ on local data $W \leftarrow w_l^{(k)}, N \leftarrow n_k$
8      end
9      Aggregate local weights to update global model:
       $w_g \leftarrow \sum_{k \in S_t} \left(\frac{n_k}{n_t}\right) \cdot w_l^{(k)}$
       Broadcast global weights $w_g$ to all clients;
10 end
11 return $w_g$
```

Table 2 shows the advantages of leveraging the P2P DFL rather than conventional CFL.

Table 2: Comparison between CFL and P2P DFL

Metric	Centralized federated learning (CFL)	P2P distributed federated learning (DFL)
Reliability	Dependent on a central server; single point of failure	Eliminates central server; no single point of failure
Availability	Limited by the central server's availability	Increased by distributing workloads across multiple nodes
Security	Central server aggregates model, increasing risk	Enhanced privacy, no single entity has access to all models
Bandwidth cost	High communication cost to and from central server	Reduced cost by leveraging local communications and private networks

(Continued)

Table 2 (continued)

Metric	Centralized federated learning (CFL)	P2P distributed federated learning (DFL)
Core network's occupancy	High occupancy due to centralized processing	Offloads processing to the network edge, reducing core occupancy

3.2 Clustered Distributed Federated Learning

In fully distributed P2P FL, two pivotal challenges emerge: scalability and non-IIDness. As the number of clients increases, the overhead associated with communication and computation escalates, resulting in prolonged convergence times. This scalability issue can severely hinder the practical deployment of FL in large-scale environments [23]. Moreover, non-IIDness—characterized by varying levels of data heterogeneity among clients—deteriorates the learning process [43], often leading to suboptimal model performance and uneven learning progress across the network.

To address these challenges, we propose a clustering approach that partitions a large community of clients into smaller groups based on selected criteria. As illustrated in Fig. 4, a cluster that initially contains clients with diverse dataset sizes is subdivided into smaller clusters where each cluster comprises clients with similar dataset sizes. In our investigation, we consider physical proximity as one grouping criterion and additionally explore clustering based on data heterogeneity. The benefits of clustering include:

1. **Improved Scalability:** Clustering significantly enhances the scalability and overall performance of FL architectures by reducing the number of clients involved in each training round.
2. **Enhanced Convergence Efficiency:** Partitioning clients into clusters according to proximity or data similarity mitigates communication overhead and accelerates convergence.
3. **Effective Handling of Non-IID Data:** Grouping clients with similar data distributions addresses non-IID challenges by ensuring that models are trained on more homogeneous data subsets.
4. **Improved Model Performance:** This approach fosters faster convergence, yields superior model accuracy, and enhances robustness.

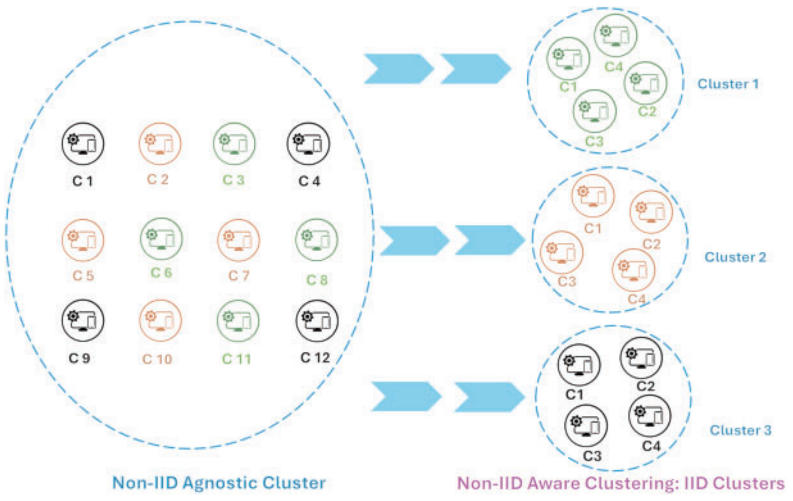

Figure 4: CDFL non-IIDness aware

The integration of CDFL with the 6G architecture is motivated by its capability to address key challenges posed by next-generation networks, including ultra-low latency, terabit-per-second data rates, massive device connectivity, and heterogeneous data distributions. Academic studies consistently highlight the synergy between FL-based approaches such as CDFL and the architectural innovations envisioned for 6G networks [44,45]. In a 6G environment, CDFL can be organized logically based on criteria such as geographic location or data distribution similarity. Recent works have leveraged complex algorithms such as federated K-means [46] or the Federated Fuzzy c-Means Algorithm (FFCM) [47]. Regardless of the specific algorithm employed, we consider two grouping strategies: vertical, where all cluster members are associated with the same edge node, and horizontal, where cluster members may belong to different edge nodes, as depicted in Fig. 5. Table 3 provides a comparative analysis between the vertical and horizontal strategies.

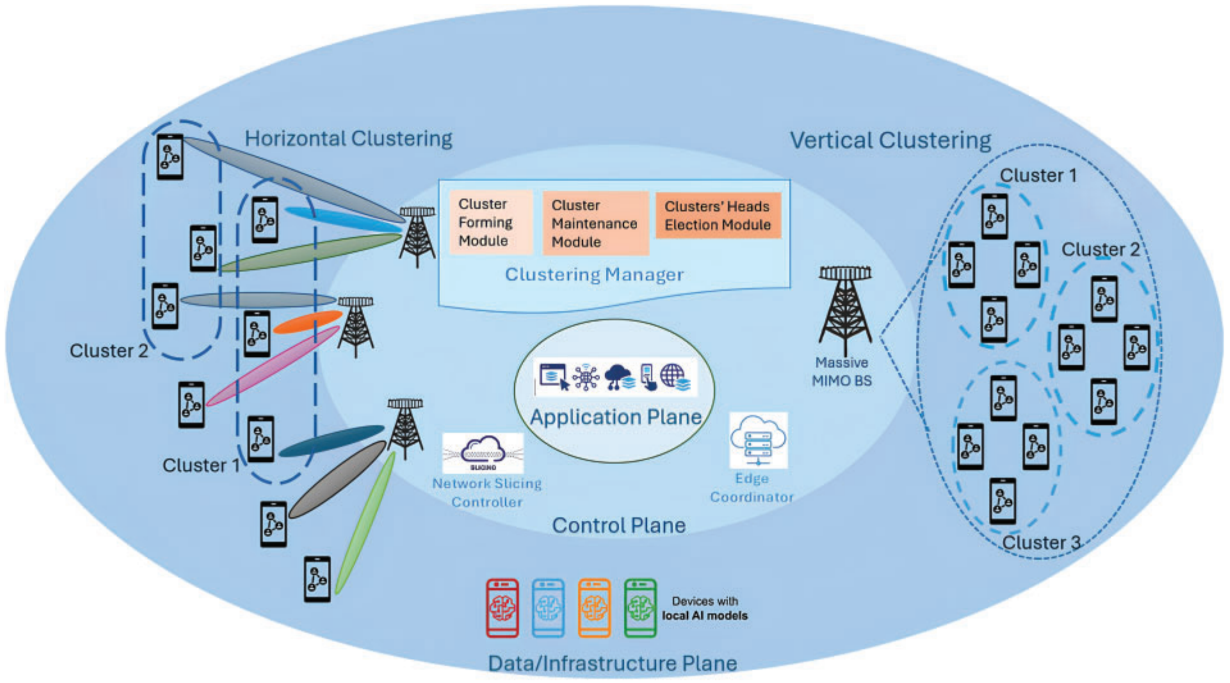

Figure 5: 6G Environment for CDFL

Table 3: Key differences between vertical and horizontal grouping strategies in CDFL

Grouping Strategy	Vertical	Horizontal
Basis	• Geography/Edge Proximity • Clients under same edge node • (e.g., factory floor robots)	• Data/Application Logic • Clients across edge nodes • (e.g., medical vs. industrial sensors)
Advantages	• Ultra-low latency (local P2P) • Minimal core network dependency • Simplified resource management	• Task-specific specialization • Scalable for heterogeneous clients • Efficient for non-IID data
Limitations	• Inflexible for mobility • Limited cross-cluster sharing • Redundant in dense deployments	• Higher coordination overhead • Needs advanced SDN • Risk of uneven cluster sizes

(Continued)

Table 3 (continued)

Grouping Strategy	Vertical	Horizontal
6G Integration	• SDN manages local topologies • URLLC slices for intra-cluster	• SDN orchestrates cross-domain • Application-tailored slices

Fig. 6 illustrates the intra-cluster and inter-cluster aggregation processes for both vertical and horizontal approaches. In the vertical scenario—where all clients are associated with the same edge node (base station)—the green circles denote the intra-cluster aggregations, while the orange circle represents the inter-cluster aggregation among cluster heads. The both aggregations are conducted using the same P2P FL process we defined in the Algorithm 3. A similar notation applies to the horizontal scenario, with the distinction that clients are associated with different edge nodes.

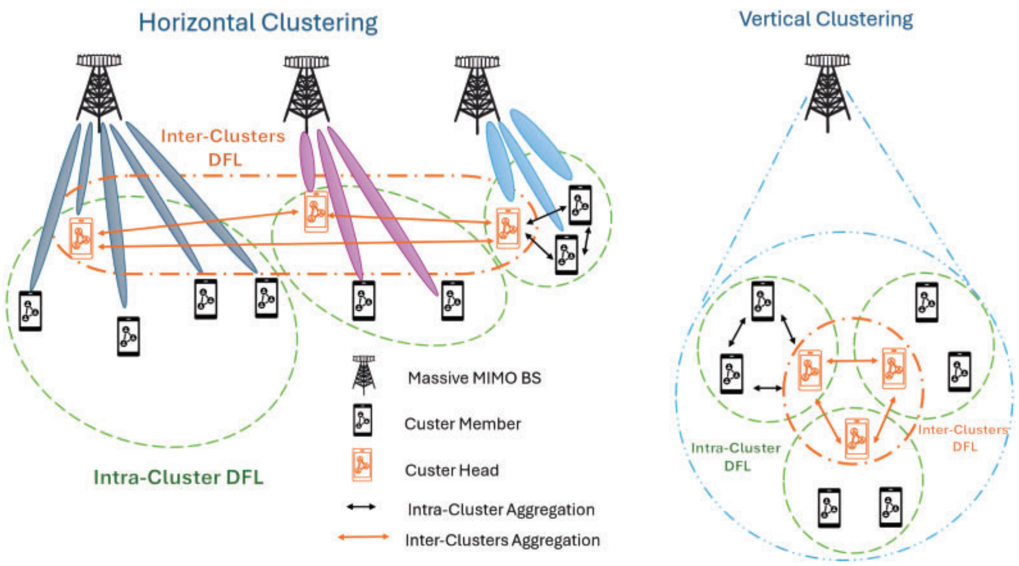

Figure 6: Intra-cluster and inter-clusters aggregation for vertical and horizontal

From software-defined networking (SDN) perspective, hardware and data storage (comprising trainers and aggregators) are allocated in the infrastructure (data plane), while the control plane comprises all coordinating and managing entities, and the application plane resides at the top. The key components involved in the clustering and aggregation processes are described below:

1. **Clustering Coordination**
   (a) *Edge Coordinator:* A Virtual Network Function (VNF) responsible for coordinating client nodes during the aggregation process. It ensures alignment among clients and orchestrates the aggregation steps under one or more edge base stations, thereby serving as the conduit to other network planes.
   (b) *Cluster Manager:* A comprehensive component for managing the lifecycle of clusters, which includes several VNFs assigned specific roles:
      i. *Cluster Construction Module:* Constructs clusters based on client data characteristics and performance metrics. It analyzes client data and locations to determine the optimal clustering strategy prior to aggregation.

ii. *Cluster Maintenance Module:* Monitors existing clusters' performance and orchestrates regrouping or reconstruction as needed.
iii. *Cluster Head Election Module:* Elects cluster heads based on criteria such as intra-cluster aggregation results, test accuracy, or resource availability.

(c) *Network Slicing Module:* Utilizes network slicing to optimize resource allocation and performance across different clusters, particularly when clients and cluster heads are associated with different edge nodes. This integration enables tailored network slices that meet the unique demands of various FL tasks while ensuring efficient network communication.

2. **Aggregation Process**
   (a) *Intra-Cluster Aggregator:* Each client node (i.e., cluster member) shares its local model with other members on a P2P basis.
   (b) *Inter-Cluster Aggregator:* During inter-cluster aggregation, cluster heads aggregate models from different clusters, ensuring that the resulting global model reflects the contributions of multiple clusters. Both aggregation processes leverage the hardware and raw data available at the client level.

For both vertical and horizontal cases, the workflow of clustering and aggregation is illustrated in Fig. 7. The flowchart outlines the proposed CDFL architecture workflow, beginning with the selection of a clustering strategy—either horizontal or vertical. In horizontal clustering, clients with similar data distributions are grouped together, followed by network slice allocation, local (intra-cluster) model aggregation, cluster head election, inter-cluster aggregation, and global model distribution. In vertical clustering, clients are grouped based on complementary features, after which intra-cluster aggregation, cluster head election, cross-cluster model aggregation, and global model dissemination are executed. Both approaches iterate until model convergence is achieved, ensuring efficient and scalable learning.

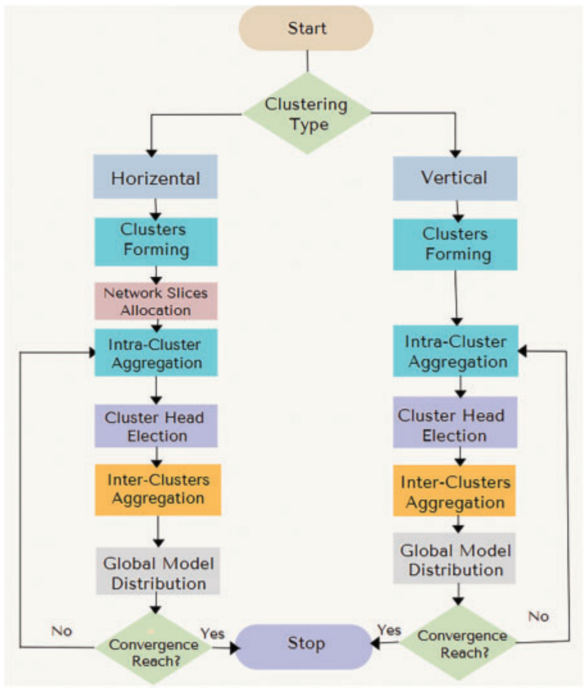

Figure 7: CDFL workflow steps

The details of intra-cluster and inter-cluster aggregation are provided below.

Intra-Cluster FL

1. Each client trains its local model for a specified number of local epochs.
2. After local training, clients in the same cluster share their local model in a P2P manner.
3. Each client aggregates its global model based on the received models.

Inter-Cluster FL

1. Cluster heads further train their models for a specified number of local epochs.
2. Cluster heads share their local models in a similar P2P way to the first phase.
3. Cluster heads aggregate their models to form a global model until they reach the required convergence.
   - The global model is returned to all clients within each cluster.

Algorithm 4 presents a two-phase training process designed to enhance model accuracy. Initially, in the intra-cluster phase, each client within a cluster trains its local model over a specified number of epochs. The resulting local models are then aggregated to form a unified cluster model.

Subsequently, cluster heads are selected from each cluster using a Multi-Strategy Fusion Snake Optimizer (MSSO) [48]. MSSO is an advanced meta-heuristic algorithm developed for distributed FL to address key challenges in inter-cluster communication. By integrating multiple optimization strategies—such as adaptive mutation, bidirectional search, and dynamic parameter updates—MSSO effectively balances the trade-off between exploration and exploitation during the cluster head selection process. This method considers both residual energy and cluster characteristics, ensuring a fair rotation of cluster heads and preventing the repetitive selection of energy-depleted nodes. Moreover, MSSO adapts to dynamic network topologies and optimizes inter-cluster separation, thereby reducing communication overhead and improving model convergence in non-IID data settings. Its flexibility in handling varying numbers of clients and incorporating newly joined nodes makes it particularly suitable for real-world FL systems with fluctuating client availability [49]. MSSO shows effectiveness in solving complex cluster heads selection problems, particularly in energy-constrained environments [48].

In the inter-cluster phase, the selected cluster heads train their models over a designated number of epochs and subsequently aggregate these models to form a global model. This global model is then disseminated back to the clients within each cluster for further updates. This two-phase approach leverages both localized data distributions and broader inter-cluster knowledge sharing, ensuring efficient learning.

Algorithm 4: Intra-cluster and inter-clusters training

```
   Output: Updated global model weights $w_g$
1  Initialize global model weights $w_g \leftarrow w_{g_0}$;
2  Intra-Cluster Phase; // Decentralized P2P within clusters
3  for round $t = 1, 2, \ldots, T_{\text{intra}}$ do
4      foreach client k in parallel do
5          for local epoch 1 to $E$ do
6              Train local model: $w_l^{(k)} \leftarrow \text{LocalUpdate}(w_g)$;
7          end
8          Share $w_l^{(k)}$ with cluster members; // P2P communication
9      end
10     foreach cluster $c \in C$ do
11         Aggregate models:
```

(Continued)

Algorithm 4 (continued)

```
                    $w_c \leftarrow \sum_{k \in c} \frac{n_k}{n_c} w_l^{(k)}$
        Select cluster head: $ch_c \leftarrow$ MSSO_Select$(c, E_{\text{res}}, \phi, prev_{CH})$;
12 $prev\_CH \leftarrow ch_c$;
13 end
14 end
15 Inter-Cluster Phase; // Cluster head collaboration
16 for round $t = 1, 2, \ldots, T_{\text{inter}}$ do
17     foreach cluster head $ch_c$ in parallel do
18         for local epoch 1 to $E_{\text{cluster}}$ do
19             Train cluster model: $w_{ch}^{(c)} \leftarrow$ LocalUpdate$(w_c)$;
20         end
21         Share $w_{ch}^{(c)}$ with other cluster heads // P2P communication
22     end
23     Aggregate cluster head models:
                    $w_g \leftarrow \sum_{c=1}^{C} \frac{n_c}{n_{\text{total}}} w_{ch}^{(c)}$
       Broadcast $w_g$ to all clients;
24 end
25 return $w_g$
26 Function MSSO_Select ($c, E_{\text{res}}, \phi, hist$):
27     Initialize candidates with fitness:
       $f = \alpha \frac{E_{\text{res}}}{E_{\text{init}}} + \beta \phi_{\text{compactness}} - \gamma \phi_{\text{separation}} - \delta \sum_{t=1}^{T} \mathbb{I}(ch_c \in hist)$
       Apply MSSO optimization; // Bidirectional search + $\alpha$-mutation
28     Return best $ch_c$ with $E_{\text{res}} > 0.4E_{\text{init}}$ and $ch_c \notin hist$
29 End Function
```

Having established the fully DFL and CDFL architectures and workflows 3, we now turn to the methodology used to validate these designs. In Section 4, we describe the tactile-grasping use case, dataset generation via the TACTO simulator, and the experimental parameters. Section 5 then presents the results of these experiments and evaluates the relative performance of centralized FL, DFL, and CDFL in terms of accuracy, convergence time, and communication/computation costs.

4 Methodology and Experimental Setup

This part describes the use case, used dataset, the simulator, and related settings.

4.1 Dataset and Simulator

There is a notable lack of ready-made datasets for tactile manipulation tasks in machine learning, as TIoT is an emerging field. To address this gap, researchers initially attempted to utilize commercially available tactile sensors; however, these sensors proved to be expensive and were primarily designed for industrial rather than research purposes. As an alternative, an open-source simulator for high-fidelity tactile sensor simulation was employed to generate realistic touch data for robot arms.

The simulator, known as TACTO [50], is capable of replicating a variety of sensor types and excels in evaluating the stability of a robot's grasp through tactile feedback. It offers several advantages, including high-resolution data, rapid rendering, and flexibility in accommodating different sensor configurations.

Nevertheless, TACTO may not perfectly capture all aspects of real-world tactile interactions and requires further validation across diverse robotic applications.

For experimental purposes, TACTO was configured to emulate a two-fingered robotic gripper equipped with BioTac SP tactile sensors. This setup captured multimodal tactile feedback (RGB, depth, and force data) during object interactions. For each grasping attempt, the simulator recorded tactile signals under varying conditions, including different object geometries (cube, cylinder, sphere), surface textures (smooth, rough), and grasp forces ranging from 0.1 to 5.0 N, thereby replicating the diversity of real-world tactile scenarios. The sensor data were sampled at 1 kHz, in alignment with TIoT latency requirements, and rendered into 128 × 128 RGB-D frames using TACTO's underlying physics engine (PyBullet). Following simulation, the raw data underwent pre-processing: depth maps were normalized to a [0, 1] range, RGB frames were converted to grayscale to reduce dimensionality, and temporal sequences were segmented into 500 ms windows (equivalent to 500 samples per grasp). The final dataset comprised 10,000 samples, partitioned into 90% training data (stratified by object class and force levels) and 10% testing data. All configuration details, including sensor parameters and simulation scripts, are publicly documented in TACTO's repository to facilitate replication. For our work, we generated a dataset using exclusively tactile data with 10,000 samples from the simulator, as the inclusion of video data did not yield significant performance improvements; this sample size was selected as an optimal trade-off between execution speed and model performance.

4.2 Use Case

FL is increasingly applied in IoT environments, especially where the tactile Internet demands ultra-low latency and high reliability. In smart healthcare, FL enables collaborative training of predictive models on distributed patient data from wearables and medical devices, preserving privacy and regulatory compliance. For example, reference [51] highlights how FL empowers decentralized analytics for personalized healthcare while maintaining data sovereignty and security. In smart cities, FL enhances both security and efficiency across distributed sensors and edge devices. Recent work by [52] introduces adaptive clustering techniques within FL to improve security and reduce communication overhead, while blockchain-based FL frameworks (see [53]) further strengthen trust and transparency in urban IoT systems. Manufacturing also benefits from FL, particularly in Industry 4.0/5.0 settings where multiple stakeholders need to collaborate without exposing proprietary data. FL allows manufacturers to develop robust models for predictive maintenance and quality control jointly [54]. In tactile Internet scenarios, advanced FL architectures enable responsive, real-time applications like remote surgery and industrial automation by leveraging multi-edge clustering and AI-driven communication strategies [55,56]. A comprehensive study of such use cases can be found in this work [19].

A notable example is a robotic arm that manipulates both virtual and physical objects based on remote operator input, thereby demonstrating the seamless integration of human control and autonomous machine learning. The tactile data used in this study are sourced from the TACTO simulator [50], a sophisticated tool designed to emulate tactile sensing environments. In our use case, the predicted tactile signals are fed directly into the robot's control loop, allowing the system to adjust its grip or motion trajectory before undesired events (such as slippage or excessive force) occur. This predictive feedback reduces reaction time and enhances task success rates, as evidenced by improved performance metrics in related studies [57,58]. Fig. 8 illustrates the integration of the tactile prediction module with the decision-making process.

The CDFL architecture is employed to train a binary classifier designed to predict the success or failure of a grasping attempt based solely on tactile sensor data. The grasping stability dataset from the TACTO simulator is used to train and evaluate the CDFL-based model. By using the grasping stability task as a concrete use case, we validate the effectiveness of the proposed CDFL architecture in enabling efficient and accurate tactile-based decision-making for real-world TIoT applications. Accurate prediction of grasping success or failure has significant implications in robotics, manufacturing, and other domains where reliable tactile sensing is crucial.

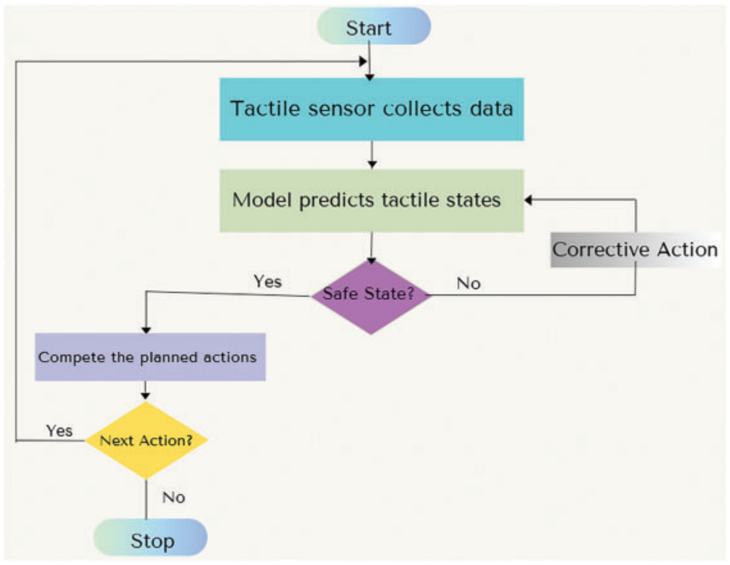
Start
Tactile sensor collects data
Model predicts tactile states
Corrective Action
Yes
Safe State?
No
Compete the planned actions
Yes
Next Action?
No
Stop

Figure 8: Decision making process

4.3 Settings

For each object grasping attempt (sample), the tactile data consisted of four attributes. This configuration results from simulating a two-fingered gripper equipped with sensors on both the right and left sides, where each side captures color and depth information as illustrated in Fig. 9.

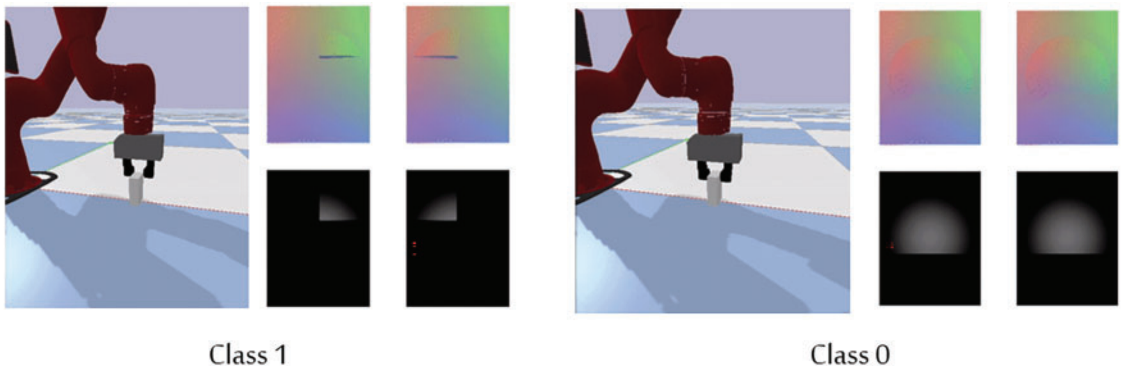

Figure 9: Grasping stability use case

After evaluating various attribute combinations, the researchers decided to focus solely on one attribute—namely, the color information from the right sensor (ColorRight). This decision reduced complexity and saved time, as incorporating additional data did not yield significant performance improvements. For model training, 90% of the data were allocated for training, with the remaining 10% reserved for testing. The experiments were executed on Google Colaboratory (Colab), a cloud-based platform that provided Python execution environments with GPU access, thereby satisfying the computational requirements of the study.

All experiments were conducted on a local desktop featuring an Intel Core i5 CPU, 8 GB of RAM, and an Solid-state drive (SSD). The system operated on Windows 10 (64-bit) and utilized Jupyter Notebook with Anaconda 3.10 and Python 3.10.9. Additionally, NVIDIA CUDA 11.7 was employed in conjunction with an NVIDIA GeForce RTX 2060 (12 GB memory, 14 Gbps clock speed) to simulate all participants sequentially, with each client using these physical resources as needed.

Regarding hyperparameters, pre-trained lightweight ResNet models were leveraged. The aggregation strategy used was FedAvg, with the Adam optimizer. The learning rate was set to 0.001, the batch size to 64, the number of local epochs to 10, and the global epochs (communication rounds) to 5. The experimental setup included 5 clients and a test data size of 10%.

5 Results and Evaluation

In this section, we detail the results and provide a comparative discussion of performance metrics between the legacy centralized FL architecture and the P2P decentralized approach for TIoT applications. We further examine the CDFL in comparison to DFL, including an analysis of the additional overhead incurred by clustering.

With the experimental setup now defined (Section 4), we proceed in this section to evaluate the three architectures. Section 5.1 compares centralized FL against fully distributed FL (DFL), and Section 5.2 examines how clustering (CDFL) mitigates scalability and non-IID challenges. Finally, Section 5.3 analyzes the computational and communication complexity of each approach.

5.1 DFL Results and Evaluation

5.1.1 Accuracy and Loss

The evaluation focused on validation performance over 50 epochs, with Fig. 10 depicting the validation accuracy curves and Fig. 11 the validation loss trends. In the CFL setup, validation accuracy climbed steadily from 0.522 to 0.944, while validation loss fell from 1.45 to 0.13. Under DFL, validation accuracy rose from 0.512 to 0.945—nearly matching the centralized curve in Fig. 10—and validation loss declined from 1.65 to 0.14 as shown in Fig. 11. The centralized FL maintains a modestly lower loss at nearly every epoch (≈0.01–0.20 gap), even as both approaches converge to almost identical accuracy by epoch 50.

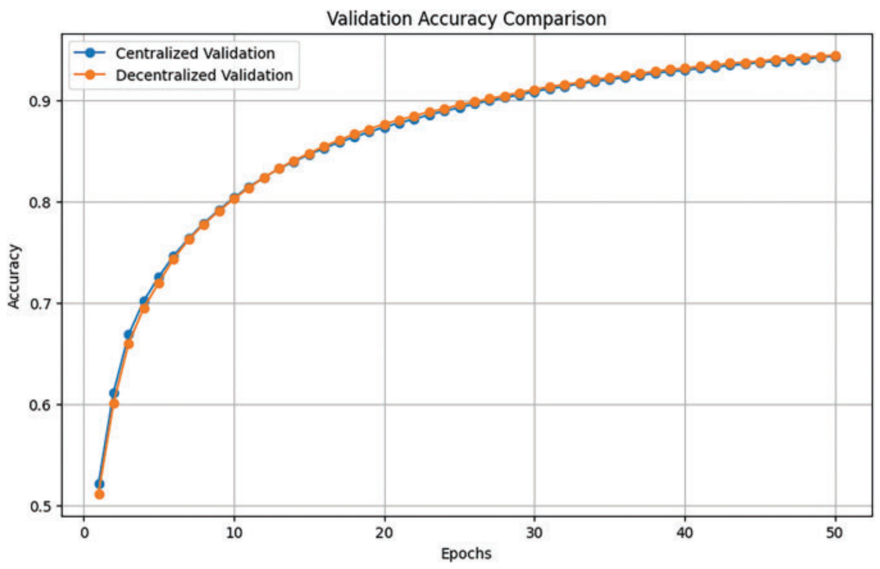

Figure 10: Accuracy comparision of centralized and decentralized FL architectures

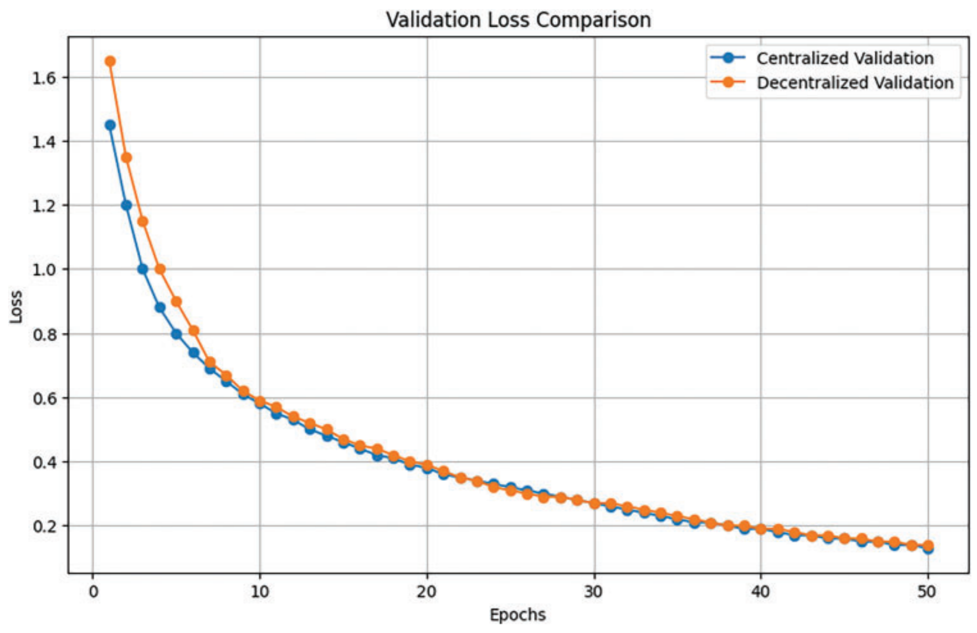

Figure 11: Loss comparision of centralized and decentralized FL architectures

These results demonstrate that DFL can achieve the same high validation accuracy as centralized FL, yet converges slightly faster in loss reduction early on. The sharper early-epoch drop in decentralized loss suggests that peer-to-peer weight exchanges act like a dynamic ensemble, delivering rapid local refinements before consensus. CFL's more gradual loss decline reflects its conservative aggregation, yielding marginally better stability and slightly lower final validation loss. Given the negligible difference in end-point accuracy (0.944 vs. 0.945) and the small loss gap, practitioners should choose between centralized and decentralized FL based on system-level priorities—scalability, privacy guarantees, and communication overhead—rather than on validation metrics alone.

5.1.2 Time to Convergence

We evaluated the convergence performance of both centralized and decentralized architectures by measuring the time and number of communication rounds required to reach 90% accuracy, as Figs. 12, and 13 illustrate. For the ResNet model, the decentralized setup achieved 90% accuracy significantly faster, with a convergence time of 6443 s compared to 8458 s in the centralized configuration, albeit requiring more communication rounds (4 vs. 1). ShuffleNet also showed improved convergence in the decentralized architecture, achieving 90% accuracy in 8369 s vs. 8736 s in the centralized case, and required fewer rounds (3 vs. 1). EfficientNet demonstrated the most dramatic improvement, with the decentralized setup converging in 5806 s compared to 10354 s for the centralized setup, while needing 2 communication rounds instead of 1. MobileNet yielded the best overall performance in both architectures, achieving convergence in 5144 s in the decentralized setup compared to 5877 s in the centralized one, though it required more rounds (3 vs. 1).

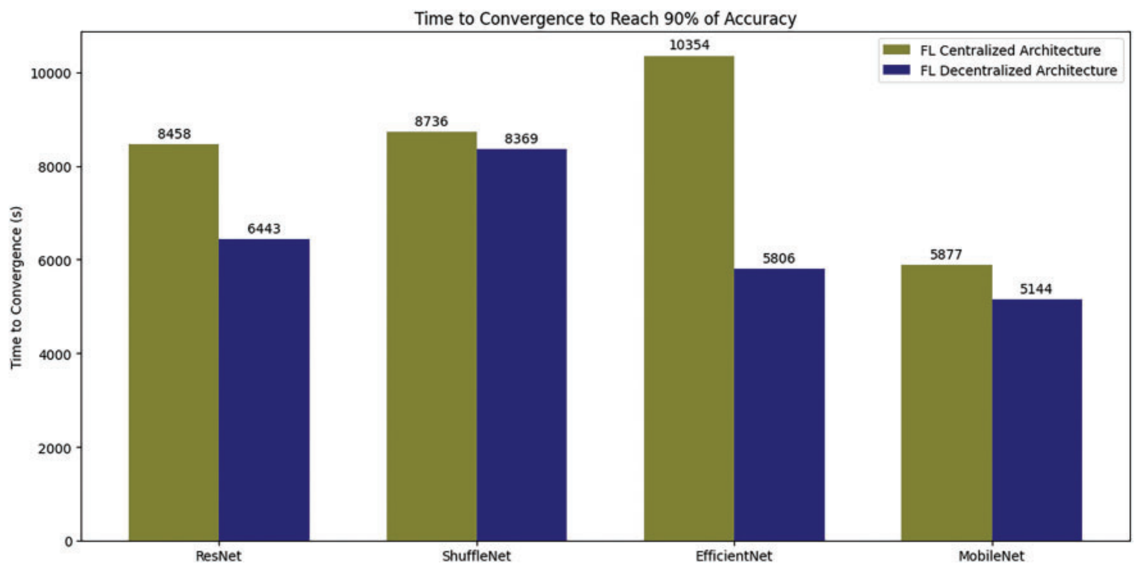

Figure 12: Time to convergence comparison of centralized and decentralized FL architectures

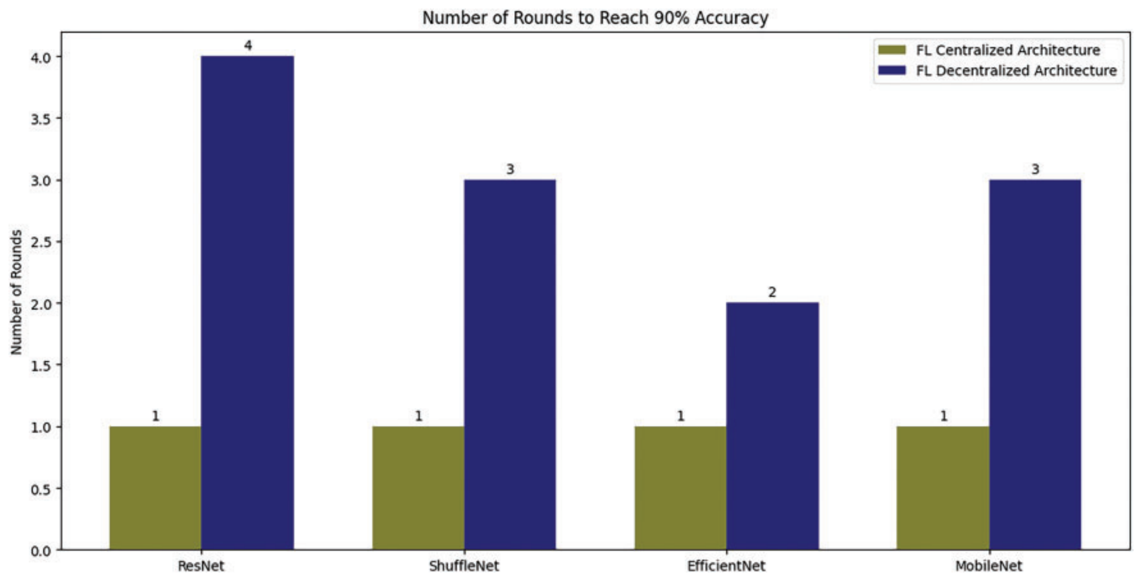

Figure 13: Number of rounds comparison of centralized and decentralized FL architectures

These results highlight that decentralized architectures can significantly reduce convergence time for reaching 90% accuracy across a range of deep learning models. This advantage is particularly pronounced in models such as EfficientNet, which exhibited nearly a two-fold decrease in convergence time. However, this reduction in time comes at the cost of an increased number of communication rounds. This trade-off may be due to the characteristics of decentralized training, where peer-to-peer communication and data distribution accelerate individual iterations but necessitate additional rounds for synchronization and model convergence. These findings underscore the potential of decentralized architectures in minimizing convergence time and suggest that further optimization of communication protocols and synchronization strategies could enhance the overall efficiency and practicality of decentralized FL systems.

5.1.3 Impact of Aggregation with Different Number of Clients

In a standard distributed architecture, local models from all clients are typically aggregated to form a global model, enabling the system to leverage the full diversity and richness of the combined dataset. However, an alternative strategy involves selecting only a subset of clients for aggregation, allowing for a trade-off between communication overhead and model performance. Fig. 14 presents the results for a single cluster containing 12 clients and a dataset of 1000 samples, evaluated under varying numbers of participating clients in each aggregation round.

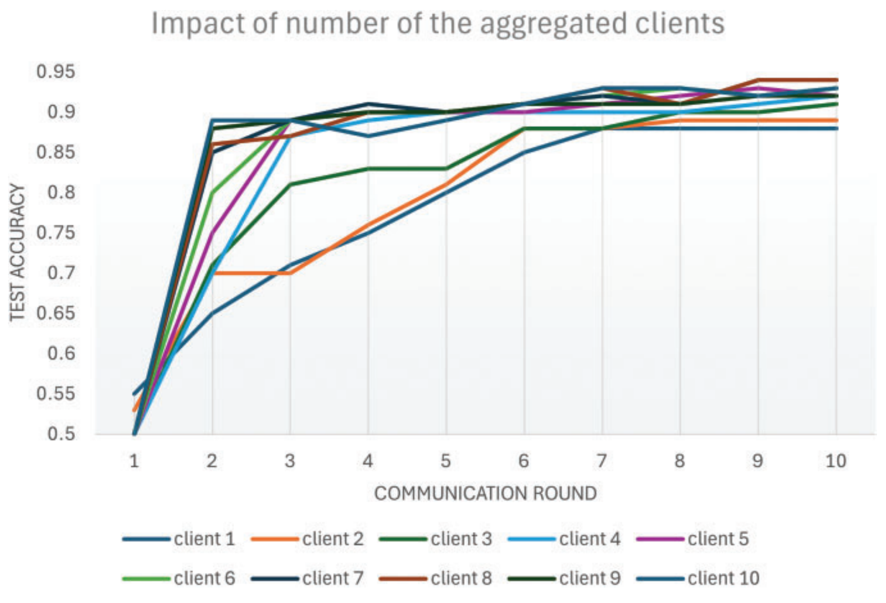

Figure 14: Impact of aggregation with different number of clients

The findings indicate that when fewer clients participate in the aggregation, more communication rounds are required to achieve the same accuracy level. Nevertheless, all configurations eventually surpassed the 85% accuracy threshold after approximately 7 rounds. This result underscores the significance of balancing communication efficiency with model performance when determining the optimal number of clients to include in each aggregation round. Furthermore, the ideal selection strategy is influenced by the heterogeneity and representativeness of the data distributed across clients. Carefully choosing clients that collectively offer a comprehensive view of the data can maintain high accuracy while reducing communication costs.

5.2 CDFL Results and Evaluation

This section presents a detailed comparison of key performance metrics between DFL and CDFL, focusing on two primary challenges: scalability and data heterogeneity (non-IIDness). The same simulation environment, dataset, and configuration described earlier are used here, although each experiment includes distinct input parameters and assumptions, introduced at the beginning of each analysis. Our focus is on the learning dynamics rather than the specifics of the clustering process; thus, we assume certain group structures and sizes throughout.

5.2.1 Scalability Problem

Scalability Performance Degradation's Impact:

Scalability in fully decentralized FL is evaluated based on the impact of increasing the number of clients on convergence performance, measured in terms of the number of global iterations (communication rounds) required to achieve a fixed test accuracy. As client count increases, so does communication overhead, potentially degrading convergence speed due to the rising frequency and complexity of model synchronization.

Fig. 15 illustrates this effect in the DFL setup with a dataset of 1000 samples, revealing a near-linear relationship between the number of clients and the time required to reach 90% test accuracy.

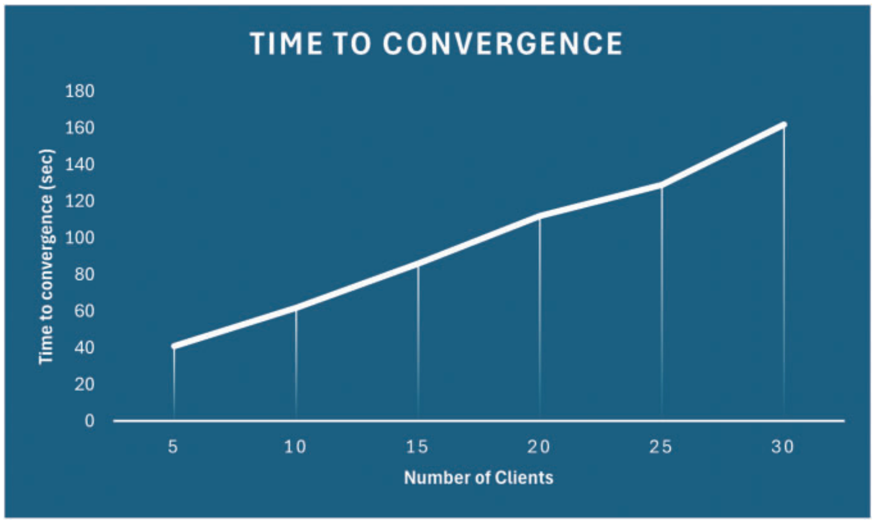

Figure 15: Impact of clients' number on the convergence

Mitigating the Scalability by Clustering:

In contrast, Fig. 16 demonstrates how CDFL effectively mitigates the scalability issue by introducing a hierarchical training structure composed of intra-cluster and inter-cluster phases, as previously described. In an experiment involving 12 clients (each with 1000 samples), we compared the time required to achieve varying levels of accuracy under both architectures.

Time was chosen as the primary comparison metric over communication rounds due to the inherent structural differences between DFL and CDFL. While DFL employs a single type of communication round across all clients, CDFL alternates between intra-cluster and inter-cluster communication phases, each contributing differently to the overall process.

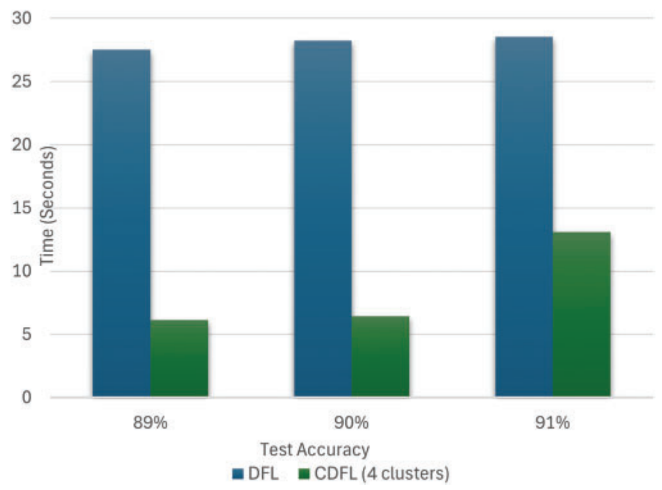

Figure 16: Time to convergence: DFL vs. CDFL

The results clearly show a significant reduction in convergence time when using the CDFL approach, highlighting its scalability benefits. Although the advantage narrows slightly at higher accuracy levels, it remains substantial, reinforcing the effectiveness of clustering in accelerating learning in large-scale, decentralized environments.

Impact of Clustering on the Communication Overhead:

Table 4, corresponding to Fig. 17, provides important insights into the trade-offs between communication overhead and accuracy in various CDFL configurations. The data reveals that increasing the number of clusters leads to a noticeable rise in communication rounds, especially in the second phase of communication.

Table 4: Communication costs for various CDFL architectures and different accuracies

Accuracy threshold	Clusters	1-Phase communication rounds	2-Phase communication rounds
	2 clusters	3	0
89%	3 clusters	5	1
	4 clusters	5	2
	2 clusters	5	0
90%	3 clusters	5	2
	4 clusters	5	3
	2 clusters	5	2
91%	3 clusters	5	3
	4 clusters	5	4

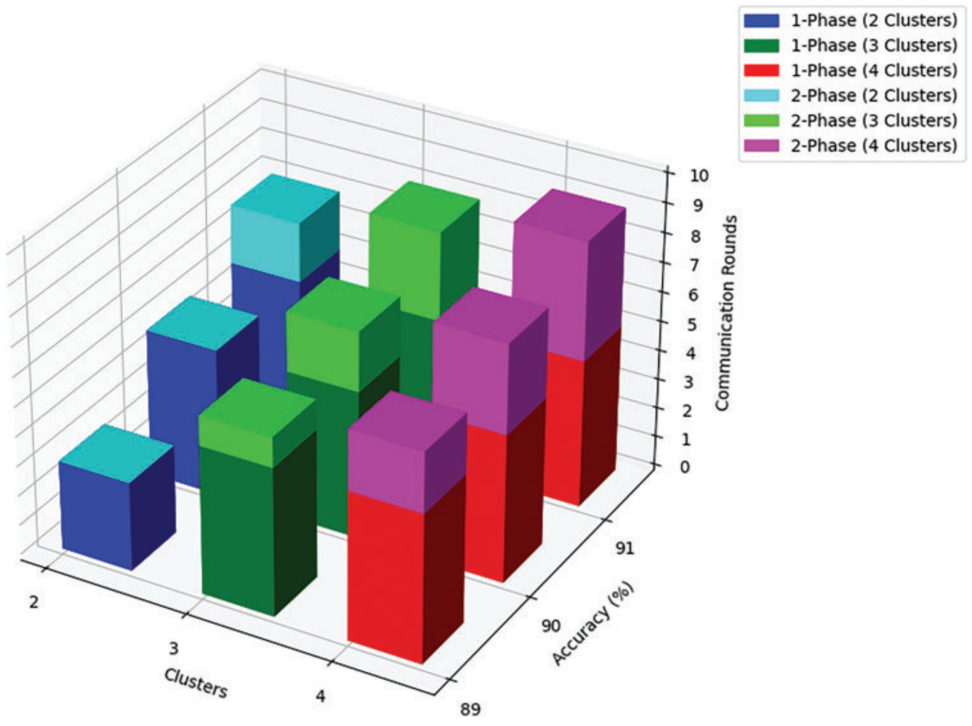

Figure 17: Communication costs for various CDFL architectures and different accuracies

This second phase—inter-cluster communication—is inherently more expensive due to longer transmission paths between cluster heads, as opposed to the intra-cluster communication of the first phase, which remains relatively low-cost due to local interactions. For example, when the number of clusters increases from 2 to 4, the second-phase communication rounds required to reach 89% accuracy rise from 0 to 2, reflecting this additional overhead.

Furthermore, a clear pattern emerges linking accuracy targets to communication cost. Achieving higher accuracy consistently demands more communication rounds. As an illustration, reaching 91% accuracy with 4 clusters necessitates 4 rounds of second-phase communication, while only 2 rounds are needed for 89% accuracy. This trend highlights the inherent trade-off between model accuracy and communication efficiency within CDFL architectures.

Increasing the number of clusters leads to a higher total number of communication rounds across both phases. While clustering enhances scalability and can accelerate convergence by distributing the training burden, it also introduces communication complexity—particularly in the inter-cluster phase. These findings underscore the need to carefully balance clustering granularity against the desired model performance and communication resource constraints.

5.2.2 Non-IIDness Problem

Proportional Bias Concept:

In practical FL scenarios, variations in client-side performance, data distributions, and computational resources result in inconsistent local training progress. Such heterogeneity can manifest as a form of *bias* when measuring the communication and computation costs. Therefore, it becomes essential to model these discrepancies systematically to achieve fair and efficient aggregation or scheduling. In this work, we focus on the heterogeneity type of the data.

We define the *bias level* of a client as a normalized measure of its deviation from a reference identical dataset. Let ψ_i denote the dataset size (number of samples) for client i, and let $\bar{\psi}$ represent the reference or average samples across all participating clients. The deviation of each client from this average is central to our bias modeling.

To represent the relative deviation proportionally, we introduce the **Proportional Bias (PB)** as follows:

$$\mathrm{PB}_i = \frac{\psi_i - \bar{\psi}}{\bar{\psi}} \tag{2}$$

Here, PB_i captures the normalized deviation of client i's cdataset size from the average size. A value of $\mathrm{PB}_i > 0$ implies that client i has a higher cost than average (i.e., more biased), while $\mathrm{PB}_i < 0$ indicates a lower-than-average size (i.e., less biased or more efficient). When $\mathrm{PB}_i = 0$, the client aligns exactly with the average.

Derivation Justification:

To contextualize this, consider the bias as a relative difference rather than an absolute one. If we were to use the absolute deviation $\psi_i - \bar{\psi}$, it would not fairly reflect the scale of differences, especially when comparing across diverse environments. Hence, dividing by the reference size $\bar{\psi}$ ensures proportional fairness and allows for meaningful comparison.

Illustrative Example:

Assume three clients with the following sizes:

$$\psi = \{120, 100, 80\} \text{ (in MB)}$$

Then,

$$\bar{\psi} = \frac{120 + 100 + 80}{3} = 100$$

Proportional bias values:

$$\mathrm{PB}_1 = \frac{120 - 100}{100} = 0.20$$
$$\mathrm{PB}_2 = \frac{100 - 100}{100} = 0.00$$
$$\mathrm{PB}_3 = \frac{80 - 100}{100} = -0.20$$

As seen above, client 1 is 20% more than average, client 3 is 20% less, and client 2 is perfectly aligned.

This proportional bias value serves as a foundational metric for subsequent weighting schemes, selection strategies, or cost-adjusted aggregation mechanisms within the proposed framework.

Impact of proportional bias on the communication overhead:

Fig. 18 illustrates the exponential relationship between the proportional bias and the number of communication rounds required for model convergence. As described earlier, the proportional bias quantifies the relative deviation of each client's dataset size from the average. A higher proportional bias implies a greater disparity, which can severely affect synchronization and model aggregation efficiency in FL.

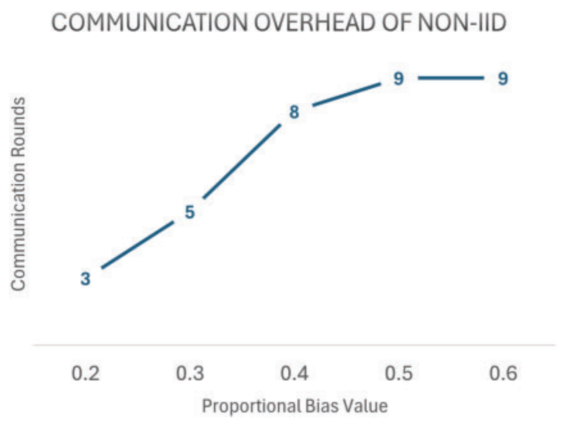

Figure 18: Impact of proportional bias on the communication overhead

Fig. 19 presents the performance degradation associated with different levels of proportional bias. It demonstrates how increased bias results in a significant drop in test accuracy and a prolonged convergence process. In extreme cases of high bias, the model may not converge within a reasonable number of rounds.

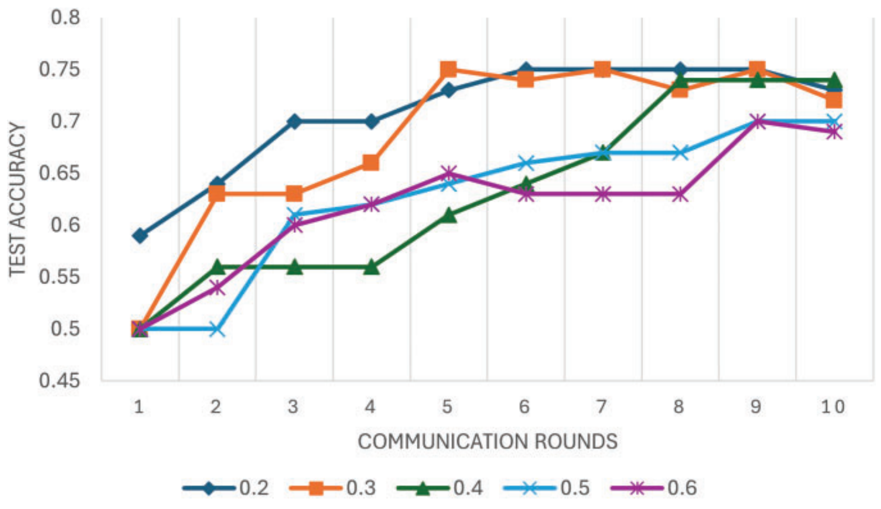

Figure 19: Impact of different proportional bias values on the convergence

In this experiment, we evaluated five distinct bias configurati ons across three levels, with a step size of 0.2 between successive levels, as detailed in Table 5. Each configuration simulates varying levels of proportional bias by adjusting the contribution share of clients to the global model. The setup involves six clients, each capable of contributing up to 2k samples (representing 100% participation). The evaluation metric is the number of communication rounds required to reach a test accuracy threshold of 70%.

Table 5: Table of bias values ξ and associated levels

Bias value	Level 1	Level 2	Level 3
ξ_1	100%	80%	60%
ξ_2	90%	70%	50%

(Continued)

Table 5 (continued)

Bias value	Level 1	Level 2	Level 3
ξ_3	80%	60%	40%
ξ_4	70%	50%	30%
ξ_5	60%	40%	20%

Mitigating the Non-IIDness performance degradation by Clustering:

This experiment addresses the performance degradation commonly caused by Non-IID data distributions by grouping clients with similar data characteristics into the same cluster, as previously described. Fig. 20 presents the test accuracy progression over communication rounds for 12 clients, each possessing 1000 data samples.

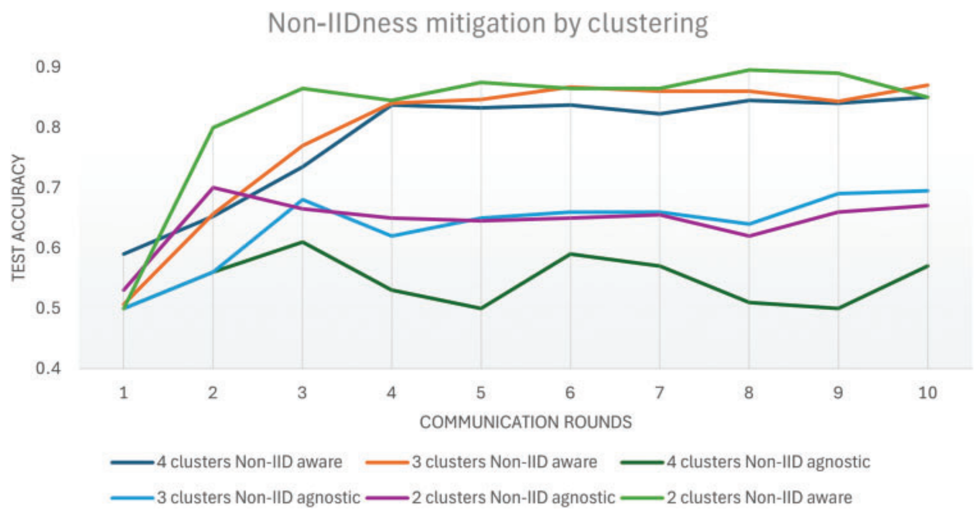

Figure 20: Mitigating the Non-IIDness performance degradation by Clustering

The results clearly demonstrate that incorporating Non-IID awareness into the architecture leads to significantly improved performance. Specifically, in the 4-cluster configuration, the Non-IID aware design achieves a smooth and consistent accuracy increase, reaching a peak of 0.85 after 10 communication rounds. In contrast, the Non-IID agnostic architecture struggles with performance instability, ultimately achieving a much lower peak accuracy of only 0.57. This contrast highlights the value of designing architectures that are cognizant of underlying data heterogeneity, as such awareness allows for better optimization and convergence.

However, the benefits of Non-IID awareness do not scale linearly with the number of clusters. The experiment reveals a slight performance decline as the number of clusters increases. For instance, the 2-cluster Non-IID aware setup achieves the highest peak accuracy at 0.89. Increasing to 3 clusters leads to a marginally lower peak of 0.87, and the 4-cluster setup further declines to 0.85. These findings suggest that while clustering similar clients enhances model accuracy under Non-IID conditions, excessively increasing the number of clusters may introduce inefficiencies. These could stem from elevated communication costs or increased complexity in coordinating across multiple clusters, which may outweigh the benefits of finer granularity in data grouping. Thus, a balance must be struck between leveraging Non-IID awareness and managing the operational overhead introduced by clustering.

Our results demonstrate that the CDFL architecture not only improves the accuracy of tactile prediction but also supports real-time decision-making by providing timely, actionable insights to the control system. This aligns with findings from recent works, which show that tactile prediction models can be directly leveraged for closed-loop control in contact-rich tasks [57,58].

5.3 Complexity Analysis

In this section, the modeling of both DFL and CDFL systems will be conducted. results and discussion of both computing and communication costs will be detailed.

5.3.1 DFL Computing and Communication Costs

Computing Cost:

The computing cost represents the total number of operations the local models perform on the clients during training. It can be defined as:

$$\text{Computing Cost} = \sum_{i=1}^{N}\sum_{e=1}^{E}\sum_{b=1}^{B_i}\text{Ops}(x_{i,e,b}) \tag{3}$$

where:

- N is the number of clients.
- E is the number of local epochs.
- B_i is the number of batches for client i.
- $\text{Ops}(x_{i,e,b})$ is the number of operations performed on the batch b in epoch e by client i.

Communication Cost:

The communication cost accounts for the total data exchanged between the server and clients during the FL process. It can be defined as:

$$\text{Communication Cost} = \sum_{r=1}^{R}\left(\sum_{i=1}^{N}\text{UploadSize}(M_r) + \sum_{i=1}^{N}\text{DownloadSize}(M_r)\right) \tag{4}$$

where:

- R is the number of global rounds.
- N is the number of clients.
- UploadSize (M_r) is the size of the model M_r uploaded by each client in round r.
- DownloadSize (M_r) is the size of the model M_r downloaded by each client in round r.

In our experiments, the model size is calculated as follows:

$$\text{Model Size (MB)} = \frac{\sum_{\text{param}}\text{param.size}\times\text{param.element\_size} + \sum_{\text{buffer}}\text{buffer.size}\times\text{buffer.element\_size}}{1024^2} \tag{5}$$

These equations help us quantify the costs associated with FL, providing a basis for evaluating and optimizing different FL strategies.

The complexity metric revealed notable differences in both computing and communication costs. In centralized architectures, ResNet required 5490 operations and had a communication cost of 2133.78 MB, whereas in decentralized architectures, it required 27,450 operations and a communication cost of 5334.46 MB. ShuffleNet under centralized architecture demanded 15,210 operations and 242.6 MB in communication, while the decentralized setup required 76,050 operations and 606.51 MB. For EfficientNet, the centralized architecture needed 16,200 operations with a communication cost of 772.9 MB, compared to 81,000 operations and 1932.25 MB for the decentralized. Finally, MobileNet required 14,220 operations and 556 MB in centralized architecture, while the decentralized setup needed 71,100 operations and 1390 MB. Figs. 21 and 22 summarized these facts.

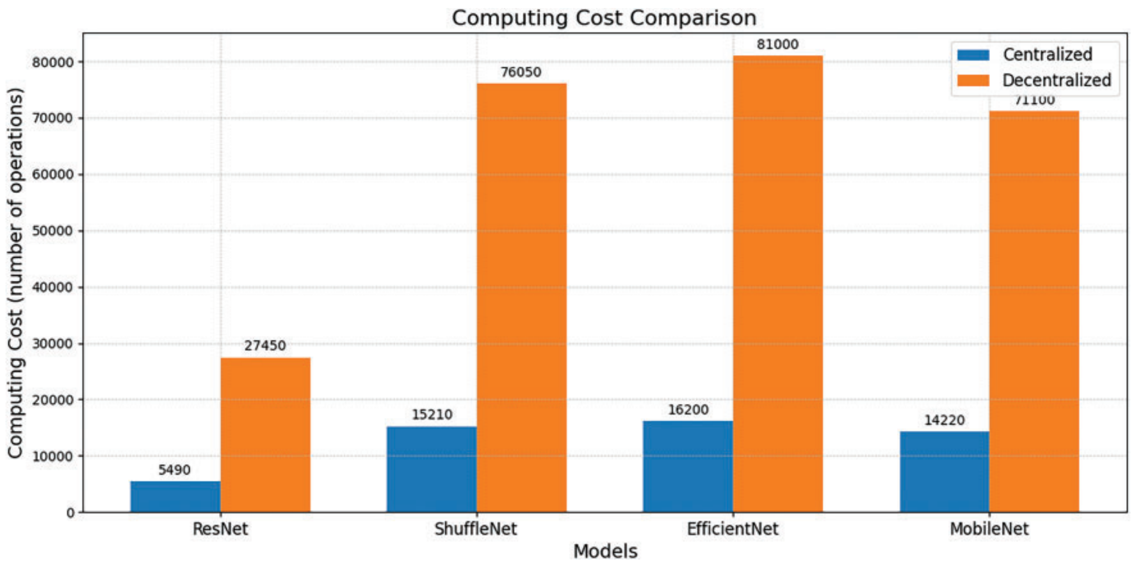

Figure 21: Computing cost comparison of centralized and decentralized FL architectures

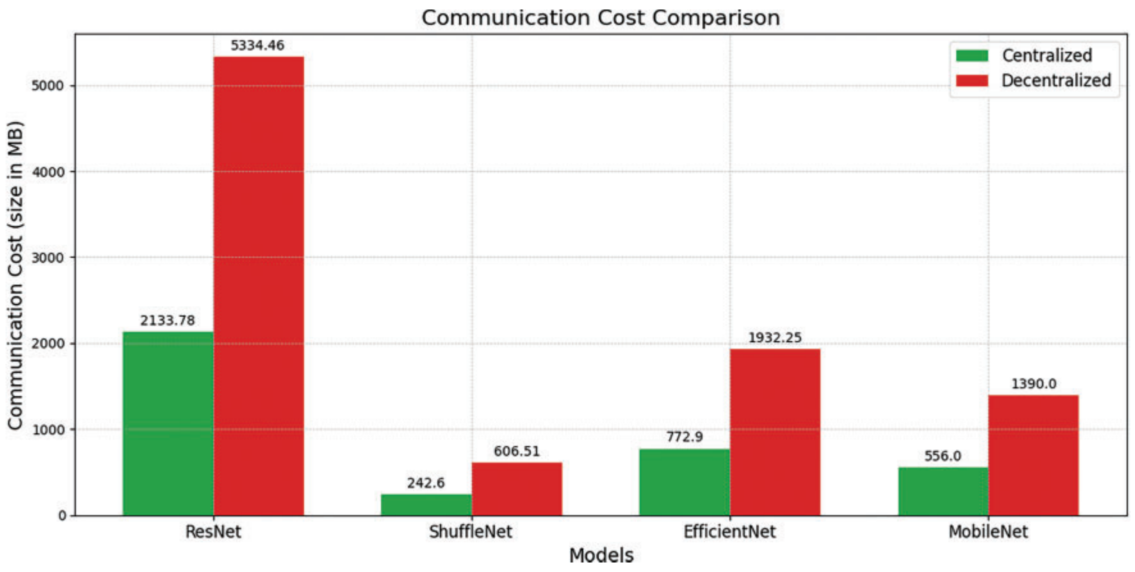

Figure 22: Communication cost comparison of centralized and decentralized FL architectures

The results indicate a significant increase in both computing and communication costs when transitioning from centralized to decentralized FL architectures. This increase is consistent across all models. For instance, ResNet and EfficientNet exhibit a fivefold increase in computing operations and more than a doubling in communication cost in the decentralized setup. This pattern is due to the inherent nature of decentralized systems, which require more frequent model updates and communication between peers, leading to higher overhead. The substantial rise in computing cost, particularly for models like ShuffleNet and MobileNet, underscores the computational inefficiency in a decentralized framework. These results suggest that while decentralized FL offers better data privacy and eliminates the need for a central server, it incurs significantly higher operational costs. Balancing these costs with the benefits of decentralized systems remains a crucial consideration for the practical deployment of FL frameworks.

5.3.2 CDFL Computing and Communication Cost

This part addresses the computing and communication costs of the CDFL and compares it with the DFL which is detailed in the first part of the experiments section.

Computing Cost:

$$\text{Computing Cost} = \left(\sum_{k=1}^{K}\sum_{i=1}^{N_k}\sum_{e=1}^{E}\sum_{b=1}^{B_i}\text{Ops}(x_{i,e,b})\right) + \left(\sum_{k=1}^{K}\sum_{e=1}^{E}\sum_{b=1}^{B_k}\text{Ops}(x_{k,e,b})\right) \tag{6}$$

where:

- K is the number of clusters.
- N_k is the number of clients in cluster k.
- E is the number of local epochs.
- B_i is the number of batches for client i.
- $\text{Ops}(x_{i,e,b})$ is the number of operations performed on batch b in epoch e by client i.
- B_k is the number of batches for the cluster leader k.
- $\text{Ops}(x_{k,e,b})$ is the number of operations performed on batch b in epoch e by cluster leader k.

Communication Cost:

$$\text{Total Communication Cost} = C_{\text{distance}} + C_{\text{cluster}} + C_{\text{intra}} + C_{\text{leader}} + C_{\text{inter}} + C_{\text{final}} \tag{7}$$

The above equation represents the six factors of communication cost of the CDFL architecture as detailed below. The used notations are illustrated in Table 6. Some terms involve downloading or uploading the model where the size of it is calculated by the Algorithm 5.

1. Distance Information Exchange

- **Variables:**
  - N: Total number of clients.
  - d: Data size exchanged between two clients during the clustering process.

  Formula:

$$C_{\text{distance}} = N \times (N-1) \times d \tag{8}$$

2. Cluster Assignment Communication

- **Variables:**

- K: Number of clusters.
- n: Number of clients per cluster (assuming equal-sized clusters, $n = \frac{N}{K}$).
- m: Size of the cluster membership information.

Formula:

$$C_{\text{cluster}} = K \times n \times (n-1) \times m \tag{9}$$

3. Intra-Cluster Communication Cost (Phase 1 of Distributed FL)

- **Variables:**
  - R: Number of global rounds.
  - N_k: Number of clients in cluster k.
  - UploadSize(M_r): Size of the model M_r uploaded by each client in round r.
  - DownloadSize(M_r): Size of the model M_r downloaded by each client in round r.
  - D_1: Smaller distance between clients within the same cluster.

Formula:

$$C_{\text{intra}} = \sum_{r=1}^{R}\sum_{k=1}^{K}\sum_{i=1}^{N_k}\sum_{j=1, j\neq i}^{N_k} (\text{UploadSize}(M_r) + \text{DownloadSize}(M_r)) \times D_1 \tag{10}$$

4. Leader (Cluster Head Selection)

- **Variables:**
  - K: Number of clusters.
  - n: Number of clients per cluster (assuming each cluster has 4 clients).
  - l: Data size exchanged per client for accuracy information.

Formula:

$$C_{\text{leader}} = K \times n \times (n-1) \times l \tag{11}$$

5. Inter-Cluster Communication Cost (Phase 2 of CDFL)

- **Variables:**
  - R: Number of global rounds.
  - D_2: Distance between different clusters.

Formula:

$$C_{\text{inter}} = \sum_{r=1}^{R}\sum_{k_1=1}^{K}\sum_{k_2=1, k_2\neq k_1}^{K} (\text{UploadSize}(M_r) + \text{DownloadSize}(M_r)) \times D_2 \tag{12}$$

6. Final Model Distribution Cost

- **Variables:**
  - M: Size of the final model.

Formula:

$$C_{\text{final}} = M \times (N-K) \times D_1 \tag{13}$$

Total Communication Cost Formula:

$$\begin{aligned}\text{Total Communication Cost} = {} & [N \times (N-1) \times d] \\ & + [K \times n \times (n-1) \times m] \\ & + \left[\sum_{r=1}^{R}\sum_{k=1}^{K}\sum_{i=1}^{N_k}\sum_{j=1, j\neq i}^{N_k} (\text{UploadSize}(M_r) + \text{DownloadSize}(M_r)) \times D_1\right] \\ & + [K \times n \times (n-1) \times l] \\ & + \left[\sum_{r=1}^{R}\sum_{k_1=1}^{K}\sum_{k_2=1, k_2\neq k_1}^{K} (\text{UploadSize}(M_r) + \text{DownloadSize}(M_r)) \times D_2\right] \\ & + [M \times (N-K) \times D_1]\end{aligned} \tag{14}$$

Table 6: Notation's definision

Notation	Name of notation
N	Total number of clients
K	Number of clusters
n	Number of clients per cluster
d	Data size exchanged between two clients
m	Size of cluster membership information
R	Number of global rounds
N_k	Number of clients in cluster k
UploadSize (M_r)	Size of model uploaded by each client in round r
DownloadSize (M_r)	Size of model downloaded by each client in round r
D_1	Smaller distance between clients within the same cluster
l	Data size exchanged per client for accuracy information
D_2	Distance between different clusters
M	Size of the final model

Algorithm 5: Calculating the model size in PyTorch

```
   Input: Predefined model global_model
   Output: Model size in megabytes (MB)
1  Initialize: param_size ← 0
2  foreach param in global_model.parameters() do
3      param_size ← param_size + param.nelement() × param.element_size()
4  Initialize: buffer_size ← 0
5  foreach buffer in global_model.buffers() do
6      buffer_size ← buffer_size + buffer.nelement() × buffer.element_size()
7  model_size_all ← (param_size + buffer_size) / $1024^2$
8  return model_size_all (in MB)
```

Fig. 23 highlights the substantial reductions in both computation and communication costs achieved through the adoption of the CDFL approach. The experiment is conducted with 12 clients, targeting a convergence threshold of 90% test accuracy. Assumptions for this scenario include communication distances of $D_1 = 0.1$ km for intra-cluster exchange, $D_2 = 1$ km for inter-cluster interactions, and an average distance $D = 0.5$ km. Additionally, the data file size for tasks such as distance exchange, cluster assignment, and cluster head selection is set to 1 KB.

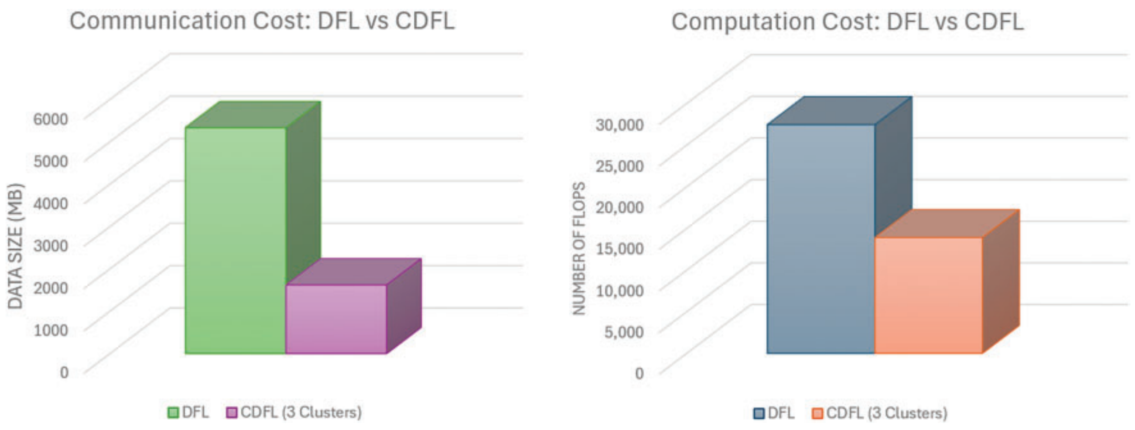

Figure 23: Communication and computing cost comparison of DFL vs. CDFL

In the traditional DFL setup, the computation cost is notably high, totaling 1,067,568 units, with a communication cost of 2932.984 units. In contrast, the CDFL approach dramatically reduces the computation cost to 139,040 units and lowers the communication cost to 1623.248 units. This comparison underscores the efficiency gains of CDFL, particularly in computation, where the reduction is significantly greater than in communication.

The pronounced decrease in computation cost can be attributed to the structure of CDFL, which partitions clients into smaller, localized clusters. This strategy minimizes redundant processing by taking advantage of intra-cluster data similarities and reducing the volume of data each node must process. Although communication cost also benefits from this clustering—mainly due to reduced long-distance interactions—the gains are relatively modest because of the continued need for coordination between cluster heads during inter-cluster rounds. Overall, the results demonstrate that CDFL offers a compelling improvement in resource efficiency, especially in computation, making it a suitable choice for scalable and resource-constrained FL environments.

5.4 *Comparison Study*

In terms of time to convergence, combined cost metric, and core network overhead; We compare:

- The legacy CFL: where all clients communicate with one central server.
- DFL: peer to peer.
- DistFL [30]: where the CFL occur first inside each cluster first then the CFL happen among cluster heads.
- CDFL: where peer to peer in both intra-cluster and inter-clusters phases.

5.4.1 Combined Cost Metric

We define a **Power Consumption Index (PCI)** to unify computing and communication costs, validated by energy measurements from IoT literature like [59], where many factors contribute. For simplicity, we assume values on average as follows:

$$\text{PCI} = \underbrace{\alpha \cdot \text{Computing Cost}}_{\text{Energy (J)}} + \underbrace{\beta \cdot \text{Communication Cost}}_{\text{Energy (J)}} \tag{15}$$

where:

$\alpha = 1 \times 10^{-10}$ J/FLOP

(energy per Floating point operations (FLOP)).

$\beta = 0.5$ J/MB

(energy per MB transmitted).

5.4.2 Core Network Overhead

Due to bidirectional client-server communication, CFL imposes a significant load on the core network. For N clients and R rounds:

$$\text{Core Traffic (MB)} = R \cdot N \cdot (\text{UploadSize} + \text{DownloadSize}) \tag{16}$$

In contrast, DFL and CDFL eliminate core network dependency by leveraging edge P2P communication.

5.4.3 Comparison Study Results

Table 7 summarizes the comparison metrics for CFL, DFL, DistFL, and CDFL.

Table 7: Comparison study: CFL vs. DFL vs. DistFL vs. CDFL

Metric	CFL	DFL	DistFL	CDFL
Time to 90% acc. (s)	8458	6443	6102	5806
Computing cost (FLOPs)	5490	27,450	10,132	13,904
Communication cost (MB)	2134	5334	3254	1623
PCI (J)	1068	2668	1628	812
Core traffic (MB)	25,605	0	17,432	0

5.4.4 Key Observations

- **DFL vs. CFL:**
  - DFL reduces convergence time by **23.8%** (8458 s → 6443 s) but increases PCI by **250%** (1068 J → 2668 J) due to P2P overhead.
  - Eliminates core network traffic (**25,605 MB → 0 MB**), critical for latency-sensitive TIoT applications.

  CDFL vs. DFL:
  - CDFL reduces PCI by **70%** and accelerates convergence by **9.9%** via clustering.
  - Retains DFL's core network independence while mitigating scalability issues.

  CDFL vs. CFL:

- Taking into account the cost of the core traffic, CDFL achieves a comparable power consumption but with **31.4% faster convergence**, **higher accuracy**, and **zero core network load**.

CDFL vs. DistFL:

- DistFL outperforms CFL and DFL in terms of energy consumption because of clustering, which significantly reduces communication cost. But still suffers from core traffic due to central server aggregation. Overall, CDFL excels in energy consumption and time to convergence.

5.5 Limitations

While our experiments rely on a controlled simulation environment, real-world deployment introduces additional challenges. These include unstable peer-to-peer links, dynamic client availability, hardware heterogeneity, and the overhead of real-time synchronization and cluster head coordination. These factors may affect performance consistency and system scalability in practical TIoT settings. Future work will address these aspects through testbed-based validation and adaptive protocol design tailored to real-world network constraints.

Regarding the dataset used, due to the limited availability of robust and publicly accessible TIoT datasets, we utilized one generated by the simulator explained earlier. After some data processing, it was made suitable for FL architectures. The conducted experiments were only for small-scale binary datasets. So, the generalization and applicability for other use cases could be limited.

6 Conclusion

We introduced a CDFL framework tailored for scalable, heterogeneity-aware TIoT. First, we showed that while fully distributed FL matches centralized FL in accuracy and loss, it boosts privacy and lowers latency by removing a central server, but at the cost of higher communication overhead and challenges with non-IID data. To overcome these drawbacks, CDFL groups clients into homogeneous clusters and applies streamlined aggregation. This design cuts network traffic, speeds up convergence, and improves model accuracy. In comparative analysis, CDFL outperforms both centralized FL and P2P FL. In comparison with CFL, it reduces convergence time by 31.4%, and eliminates core-network dependency while ensuring scalability.

Looking ahead, we plan to embed CDFL into a full end-to-end TIoT architecture with dynamic resource allocation strategies, and explore integration with SDN, blockchain, and network slicing. We also recognize emerging privacy risks introduced by clustering-such as inferring data similarity from shared updates and targeting cluster heads-and will harden the system using differential privacy, secure multi-party aggregation, and Byzantine-resilient protocols. These extensions will be crucial to balance efficiency, scalability, and trust in real-world TIoT deployments.

Acknowledgement: Not applicable.

Funding Statement: This work was supported by the Deanship of Scientific Research (DSR), King Abdulaziz University, Jeddah, under grant No. GPIP: 2040-611-2024. The authors, therefore, gratefully acknowledge DSR technical and financial support.

Author Contributions: Omar Alnajar: Writing—review and editing, Writing—original draft, Project administration, Methodology, Conceptualization. Ahmed Barnawi: Validation, Supervision. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The datasets generated and analyzed during the current study are available from the corresponding author on reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Bhattacharya P, Verma A, Tanwar S. Federated learning for internet of medical things: concepts, paradigms, and solutions. Boca Raton, FL, USA: CRC Press; 2023. 320 p.
2. Lashari MH, Batayneh W, Khokhar A. Enhancing precision in tactile internet-enabled remote robotic surgery: kalman filter approach. In: 2024 International Wireless Communications and Mobile Computing (IWCMC); 2024 May 27–31; Ayia Napa, Cyprus. p. 1189–96. doi:10.1109/iwcmc61514.2024.10592575.
3. Hung C-H, Wang K-M, Hsiao S-E. Tactile vest: virtual reality interactive shooting game accessory using microcontroller and vibration motor. In: 2024 International Conference on Consumer Electronics-Taiwan (ICCE-Taiwan); 2024 Jul 9–11; Taichung, Taiwan. p. 151–2.
4. Kumar N, Ali R. A smart contract-based robotic surgery authentication system for healthcare using 6G-Tactile Internet. Comput Netw. 2024;238:110133. doi:10.1016/j.comnet.2024.110133.
5. Hernández Gobertti FA. Immersive teleoperation of a robotic arm with grip recognition and haptic feedback [master thesis]. Valencia, Spain: Universitat Politècnica de València; 2024. 150 p.
6. Wang T, Zheng P, Li S, Wang L. Multimodal human-robot interaction for human-centric smart manufacturing: a survey. Adv Intell Syst. 2024;6(3):2300359. doi:10.1002/aisy.202300359.
7. Sengupta J, Dey D, Ferlin S, Ghosh N, Bajpai V. Accelerating tactile internet with QUIC: a security and privacy perspective. arXiv:2401.06657. 2024.
8. Selim M, Dresscher D, Abayazid M. A comprehensive review of haptic feedback in minimally invasive robotic liver surgery: advancements and challenges. Int J Med Robot. 2024;20(1):e2605. doi:10.1002/rcs.2605.
9. Philip-Kpae M, Zhang Y, Kumar A, Lee J, Hassan M, Chen L, et al. 5G communication network: a comprehensive review as a cutting-edge technology in communication systems for sustainable development and future direction. Int J Eng Modern Technol. 2025;12(1):45–67. doi:10.1234/ijemt.2025.012345.
10. Huang S, Sheng B, Ji C. Channel coding at finite blocklength and its application in 6G URLLC. J Comput Commun. 2023;11(9):132–42. doi:10.4236/jcc.2023.119008.
11. Ficco M, Guerriero A, Milite E, Palmieri F, Pietrantuono R, Russo S. Federated learning for IoT devices: enhancing TinyML with on-board training. Inf Fusion. 2024;104(3):102189. doi:10.1016/j.inffus.2023.102189.
12. Hamdi W, Ksouri C, Bulut H, Mosbah M. Network slicing based learning techniques for IoV in 5G and beyond networks. IEEE Commun Surv Tutor. 2024;26(1):1–30. doi:10.1109/COMST.2024.3351234.
13. Alwakeel AM, Alnaim AK. Network slicing in 6G: a strategic framework for IoT in smart cities. Sensors. 2024;24(13):4254. doi:10.3390/s24134254.
14. Zormati MA, Lakhlef H, Ouni S. Review and analysis of recent advances in intelligent network softwarization for the Internet of Things. Comput Netw. 2024;241(1):110215. doi:10.1016/j.comnet.2024.110215.
15. Cañete A, Amor M, Fuentes L. HADES: an NFV solution for energy-efficient placement and resource allocation in heterogeneous infrastructures. J Netw Comput Appl. 2024;221:103764. doi:10.1016/j.jnca.2024.103764.
16. Appiah I, Jiang X, Boahen EK, Owusu E. A 5G perspective of an SDN-based privacy-preserving scheme for IoT networks. Int J Commun Netw Syst Sci. 2023;16(8):169–90. doi:10.4236/ijcns.2023.168012.
17. Groshev M, Baldoni G, Cominardi L, de la Oliva A, Gazda R. Edge robotics: are we ready? An experimental evaluation of current vision and future directions. Digit Commun Netw. 2023;9(1):166–74. doi:10.1016/j.dcan.2022.01.004.
18. Sáez-de-Cámara X, Flores JL, Arellano C, Urbieta A, Zurutuza U. Clustered federated learning architecture for network anomaly detection in large scale heterogeneous IoT networks. Comput Secur. 2023;131(4):103299. doi:10.1016/j.cose.2023.103299.
19. Quy VK, Nguyen DC, Van Anh D, Quy NM. Federated learning for green and sustainable 6G IIoT applications. Internet Things. 2024;25(2):101061. doi:10.1016/j.iot.2024.101061.

20. Yamany W, Keshk M, Moustafa N, Turnbull B. Swarm optimisation-based federated learning for the cyber resilience of internet of things systems against adversarial attacks. IEEE Trans Consum Electron. 2023;69(3):123–35. doi:10.1109/TCE.2023.3319039.
21. Asad M, Shaukat S, Hu D, Wang Z, Javanmardi E, Nakazato J, et al. Limitations and future aspects of communication costs in federated learning: a survey. Sensors. 2023;23(17):7358. doi:10.3390/s23177358.
22. Beltrán ETM, Gómez ÁLP, Feng C, Sánchez PMS, Bernal SL, Bovet G, et al. Fedstellar: a platform for decentralized federated learning. Expert Syst Appl. 2024;242:122861.
23. Zhou Y, Shi Y, Zhou H, Wang J, Fu L, Yang Y. Toward scalable wireless federated learning: challenges and solutions. IEEE Internet Things Magaz. 2023;6(4):10–6. doi:10.1109/iotm.001.2300099.
24. Ye M, Fang X, Du B, Yuen PC, Tao D. Heterogeneous federated learning: state-of-the-art and research challenges. ACM Comput Surv. 2023;56(3):1–44. doi:10.1145/3603621.
25. Alnajar O, Barnawi A. TactiFlex: a Federated learning-enhanced in-content aware resource allocation flexible architecture for Tactile IoT in 6G networks. Eng Appl Artif Intell. 2024;136(8):108934. doi:10.1016/j.engappai.2024.108934.
26. Zhou Y, Shi M, Tian Y, Ye Q, Lv J. DeFTA: a plug-and-play peer-to-peer decentralized federated learning framework. Inf Sci. 2024;670(4):120582. doi:10.1016/j.ins.2024.120582.
27. Behera MR, Upadhyay S, Otter R, Shetty S, Sanford BJ, Moran SJ, et al. Federated learning using peer-to-peer network for decentralized orchestration of model weights. *Authorea Preprints* [Interent]. 2023 [cited 2025 Jun 8]. Available from: https://www.authorea.com/users/663510/articles/676447-federated-learning-using-peer-to-peer-network-for-decentralized-orchestration-of-model-weights.
28. Liu J, Huo Y, Qu P, Sun X, Liu Z, Ma Q, et al. FedCD: a hybrid federated learning framework for efficient training with IoT devices. IEEE Internet Things J. 2024;11(11):20040–50. doi:10.1109/jiot.2024.3368216.
29. Gupta V, Luqman A, Chattopadhyay N, Chattopadhyay A, Niyato D. TravellingFL: communication efficient peer-to-peer federated learning. IEEE Trans Vehicular Technol. 2024;73(1):1–15. doi:10.1109/TVT.2023.3332898.
30. Liu B, Cai Y, Zhang Z, Li Y, Wang L, Li D, et al. DistFL: distribution-aware federated learning for mobile scenarios. Proc ACM Interactive Mobile Wearable Ubiquitous Technol. 2021;5(4):1–26.
31. Liu S, Liu Z, Xu Z, Liu W, Tian J. Hierarchical decentralized federated learning framework with adaptive clustering: bloom-filter-based companions choice for learning non-IID data in IoV. Electronics. 2023;12(18):3811. doi:10.3390/electronics12183811.
32. Wu Q, Chen X, Ouyang T, Zhou Z, Zhang X, Yang S, et al. Communication-efficient hierarchical federated learning with adaptive staleness control and heterogeneity-aware client-edge association. IEEE Trans Parallel Distrib Syst. 2023;34(5):1560–79. doi:10.1109/tpds.2023.3238049.
33. Chen Z, Liao W, Tian P, Wang Q, Yu W. A fairness-aware peer-to-peer decentralized learning framework with heterogeneous devices. Future Internet. 2022;14(5):138. doi:10.3390/fi14050138.
34. Solat F, Kim TY, Lee J. A novel group management scheme of clustered federated learning for mobile traffic prediction in mobile edge computing systems. J Commun Netw. 2023;25(4):1–12. doi:10.23919/JCN.2023.000023.
35. Wang J, Yang B, Li W, Zhang Z. A clustered federated learning paradigm with model ensemble in O-RAN. In: 2024 IEEE Wireless Communications and Networking Conference (WCNC); 2024 Apr 21–24; Dubai, United Arab Emirates. p. 1–6.
36. Morafah M, Vahidian S, Wang W, Lin B. Flis: clustered federated learning via inference similarity for non-IID data distribution. IEEE Open J Comput Soc. 2023;4(2):109–20. doi:10.1109/ojcs.2023.3262203.
37. Zhao Z, Wang J, Hong W, Quek TQS, Ding Z, Peng M, et al. Ensemble federated learning with non-IID data in wireless networks. IEEE Trans Wirel Commun. 2024 Apr;23(4):3557–71. doi:10.1109/TWC.2024.3447833.
38. Chen Q, Wang Z, Zhou Y, et al. CFL: cluster federated learning in large-scale peer-to-peer networks. In: Information Security: 25th International Conference, ISC 2022; 2022 Dec 18–22; Bali, Indonesia. p. 464–72. doi:10.1007/978-3-031-22390-7_27.
39. Wang Z, Xu H, Liu J, Xu Y, Huang H, Zhao Y, et al. Accelerating federated learning with cluster construction and hierarchical aggregation. IEEE Trans Mob Comput. 2023 Jul;22(7):3805–22. doi:10.1109/TMC.2022.3147792.

40. Ouyang X, Xie Z, Zhou J, Xing G, Huang J. ClusterFL: a clustering-based federated learning system for human activity recognition. ACM Trans Sens Netw. 2022;19(1):1–32. doi:10.1145/3554980.
41. Shlezinger N, Rini S, Eldar YC. The communication-aware clustered federated learning problem. In: 2020 IEEE International Symposium on Information Theory (ISIT); 2020 Jun 21–26; Los Angeles, CA, USA. p. 2610–5.
42. Du R, Xu S, Zhang R, Xu L, Xia H. A dynamic adaptive iterative clustered federated learning scheme. Knowl Based Syst. 2023;276(4):110741. doi:10.1016/j.knosys.2023.110741.
43. Zhang X, Sun W, Chen Y. Tackling the non-IID issue in heterogeneous federated learning by gradient harmonization. IEEE Signal Process Lett. 2024;31:1–5. doi:10.1109/LSP.2024.3356789.
44. Sandeepa C, Zeydan E, Samarasinghe T, Liyanage M. Federated learning for 6G networks: navigating privacy benefits and challenges. IEEE Open J Commun Soc. 2024;6:90–129. doi:10.1109/OJCOMS.2024.3358736.
45. Driss MB, Sabir E, Elbiaze H, Saad W. Federated learning for 6G: paradigms, taxonomy, recent advances and insights. arXiv:2312.04688. 2023.
46. Yang K, Mohammadi Amiri M, Kulkarni SR. Greedy centroid initialization for federated K-means. Knowl Inf Syst. 2024;66(6):3393–425. doi:10.1007/s10115-024-02066-x.
47. Stallmann M, Wilbik A. Towards federated clustering: a federated fuzzy c-means algorithm (FFCM). arXiv:2201.07316. 2022.
48. Yang L, Zhang D, Li L, He Q. Energy efficient cluster-based routing protocol for WSN using multi-strategy fusion snake optimizer and minimum spanning tree. Sci Rep. 2024;14(1):16786. doi:10.21203/rs.3.rs-3901967/v1.
49. Yang W, Chen B, Shen Y, Liu J, Yu L. Cross-fusion rule for personalized federated learning. arXiv:2302.02531. 2023.
50. Wang S, Lambeta M, Chou PW, Calandra R. TACTO: a fast, flexible, and open-source simulator for high-resolution vision-based tactile sensors. IEEE Robot Autom Lett. 2022;7(2):3930–7. doi:10.1109/lra.2022.3146945.
51. Li H, Li C, Wang J, Yang A, Ma Z, Zhang Z, et al. Review on security of federated learning and its application in healthcare. Future Gener Comput Syst. 2023;144(4):271–90. doi:10.1016/j.future.2023.03.014.
52. Zhang Y, Chen H, Lin Z, Chen Z, Zhao J. FedAC: an adaptive clustered federated learning framework for heterogeneous data. arXiv:2403.16460. 2024.
53. Moore E, Imteaj A, Hossain MZ, Rezapour S, Amini MH. Blockchain-empowered cyber-secure federated learning for trustworthy edge computing. IEEE Trans Artif Intell.2025 Jan;4(1):1–9. doi:10.1109/TAI.2025.3539258.
54. Islam F, Raihan AS, Ahmed I. Applications of federated learning in manufacturing: identifying the challenges and exploring the future directions with Industry 4.0 and 5.0 visions. arXiv:2302.13514. 2023.
55. Mughal FR, He J, Das B, Dharejo FA, Zhu N, Khan SB, et al. Adaptive federated learning for resource-constrained IoT devices through edge intelligence and multi-edge clustering. Sci Rep. 2024;14(1):28746. doi:10.1038/s41598-024-59364-1.
56. Rane J, Mallick SK, Kaya O, Rane NL. Federated learning for edge artificial intelligence: enhancing security, robustness, privacy, personalization, and blockchain integration in IoT. Fut Res Oppor Artif Intell Ind. 2024;4:93–135. doi:10.70593/978-81-981271-0-5_3.
57. Funk N, Chen C, Schneider T, Chalvatzaki G, Calandra R, Peters J. On the importance of tactile sensing for imitation learning: a case study on robotic match lighting. arXiv:2504.13618. 2025.
58. Xue H, Ren J, Chen W, Zhang G, Fang Y, Gu G, et al. Reactive diffusion policy: slow-fast visual-tactile policy learning for contact-rich manipulation. arXiv:2503.02881. 2025.
59. Yang Z, Adamek K, Armour W. Double-exponential increases in inference energy: the cost of the race for accuracy. arXiv:2412.09731. 2024.